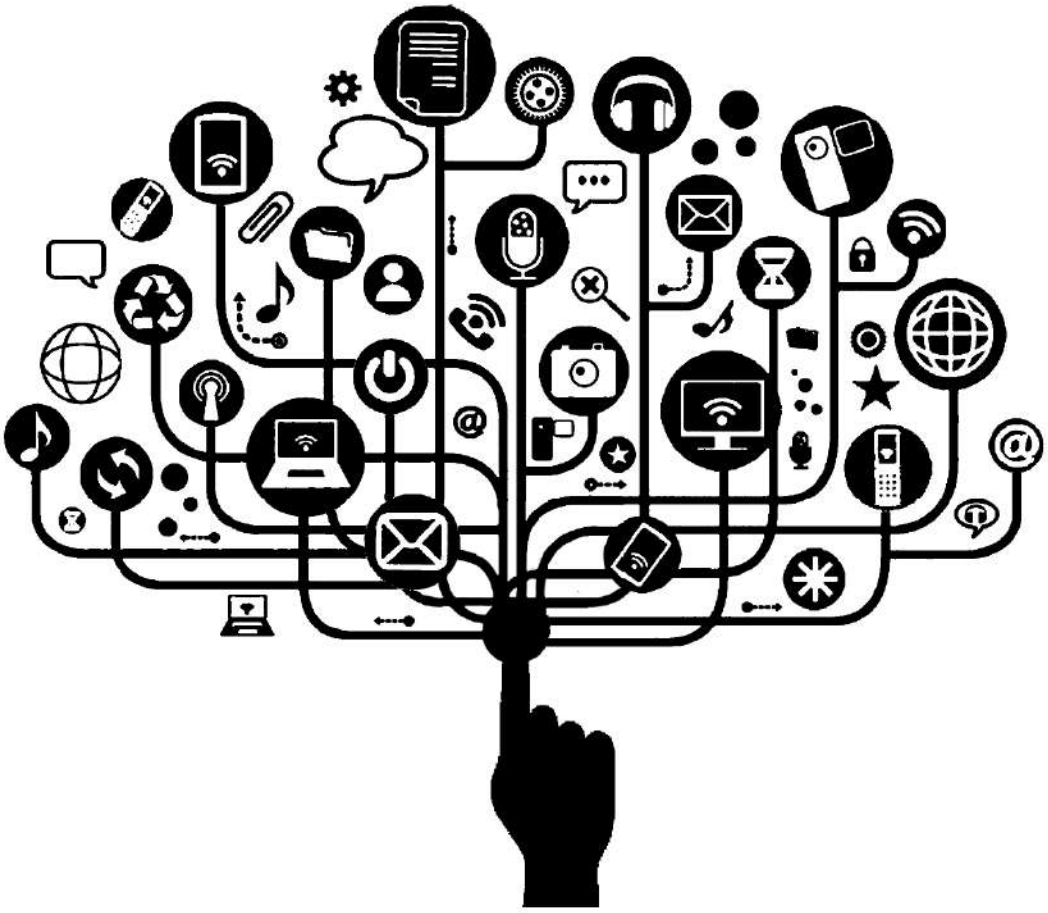
信息技术

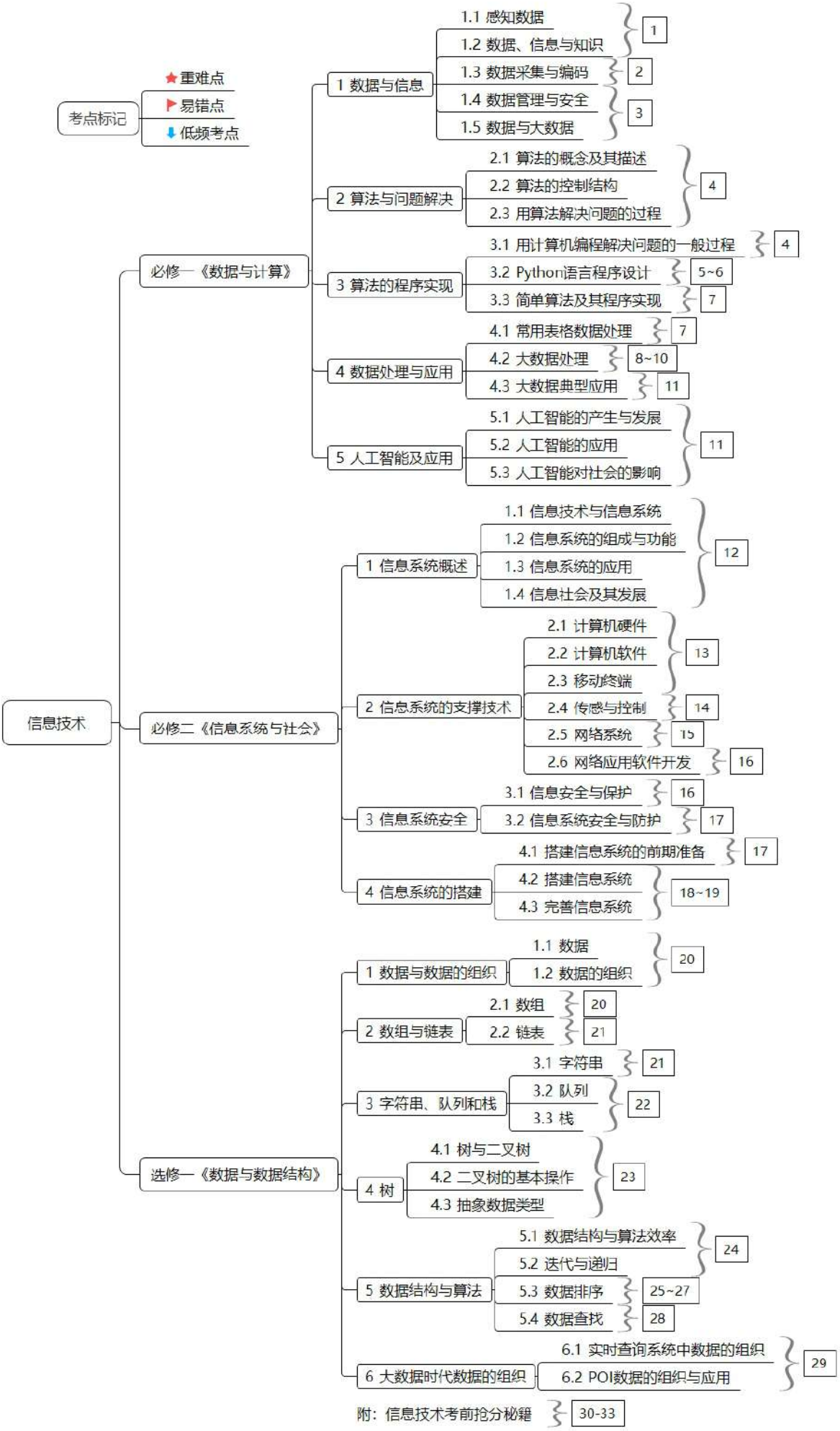
思维导图手册

读薄课本 厘清知识



浙江省新昌中学 信息技术组 陈钿歌 编写

2024.1 第5版


信息技术
考点标记
重难点
易错点
低频考点
必修一《数据与计算》
1 数据与信息
1.1 感知数据
1.2 数据、信息与知识
1.3 数据采集与编码
1.4 数据管理与安全
1.5 数据与大数据
1
2
3
2 算法与问题解决
2.1 算法的概念及其描述
2.2 算法的控制结构
2.3 用算法解决问题的过程
4
3 算法的程序实现
3.1 用计算机编程解决问题的一般过程
3.2 Python语言程序设计
3.3 简单算法及其程序实现
4
5~6
7
4 数据处理与应用
4.1 常用表格数据处理
4.2 大数据处理
4.3 大数据典型应用
7
8~10
11
5 人工智能及应用
5.1 人工智能的产生与发展
5.2 人工智能的应用
5.3 人工智能对社会的影响
11
必修二《信息系统与社会》
1 信息系统概述
1.1 信息技术与信息系统
1.2 信息系统的组成与功能
1.3 信息系统的应用
1.4 信息社会及其发展
12
2 信息系统的支撑技术
2.1 计算机硬件
2.2 计算机软件
2.3 移动终端
2.4 传感与控制
2.5 网络系统
2.6 网络应用软件开发
13
14
15
16
3 信息系统安全
3.1 信息安全与保护
3.2 信息系统安全与防护
16
17
4 信息系统的搭建
4.1 搭建信息系统的前期准备
4.2 搭建信息系统
4.3 完善信息系统
17
18~19
选修一《数据与数据结构》
1 数据与数据的组织
1.1 数据
1.2 数据的组织
20
2 数组与链表
2.1 数组
2.2 链表
20
21
3 字符串、队列和栈
3.1 字符串
3.2 队列
3.3 栈
21
22
4 树
4.1 树与二叉树
4.2 二叉树的基本操作
4.3 抽象数据类型
23
5 数据结构与算法
5.1 数据结构与算法效率
5.2 迭代与递归
5.3 数据排序
5.4 数据查找
24
25~27
28
6 大数据时代数据的组织
6.1 实时查询系统中数据的组织
6.2 POI数据的组织与应用
29
附：信息技术考前抢分秘籍
30-33

必修一《数据与计算》

1.1 感知数据

- 信息记录离不开数据，数据自古就有且无处不在
- 人类生活离不开数据，人们在利用数据的同时，自身的行为也在产生数据
- 科学研究离不开数据，数据的客观性正好为科学研究提供了可靠的依据

1.2 数据、信息与知识

数据Data

know-something

- 数据是对客观事物的符号表示
- 单纯的数据是没有意义的，经过解释，数据才变得有意义
- 表现形式可以是文字、图形、图像、音频、视频等

信息Information

know-what

- 定义
  - 信息是用来消除随机不确定性的东西（香农）
  - 信息是数据经过存储、分析及解释后所产生的意义　　数据就是信息的载体
- 特征
  - 载体依附性
    - 信息是不能独立存在的，必须依附于一定的载体
    - 同一信息可以依附于不同的载体
    - 信息可以脱离它所反映的事物被存储和传播
  - 时效性　信息会随着时间的推移而变化
  - 共享性　信息不会像物质、能源那样发生损耗，但也会消失
  - 可加工处理性
    - 形态变化
      - 数字信号和模拟信号
      - 信息的采集、编码、解码、发布
    - 真伪性
  - 价值性
    - 显性价值与隐形价值
    - 信息的价值是相对的
    - 加工后价值的变化

知识Knowledge

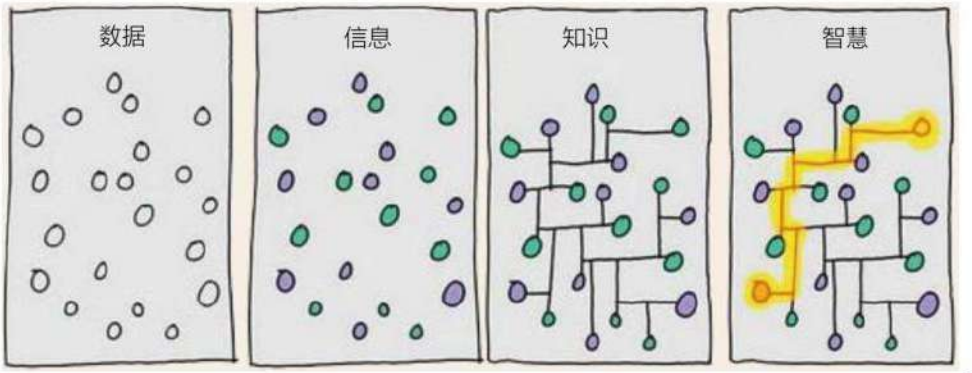
know-why

- 知识是人类在社会实践中所获得的的认识和经验的总和
- 知识是可以继承和传递的　　信息的共享性

智慧Wisdom

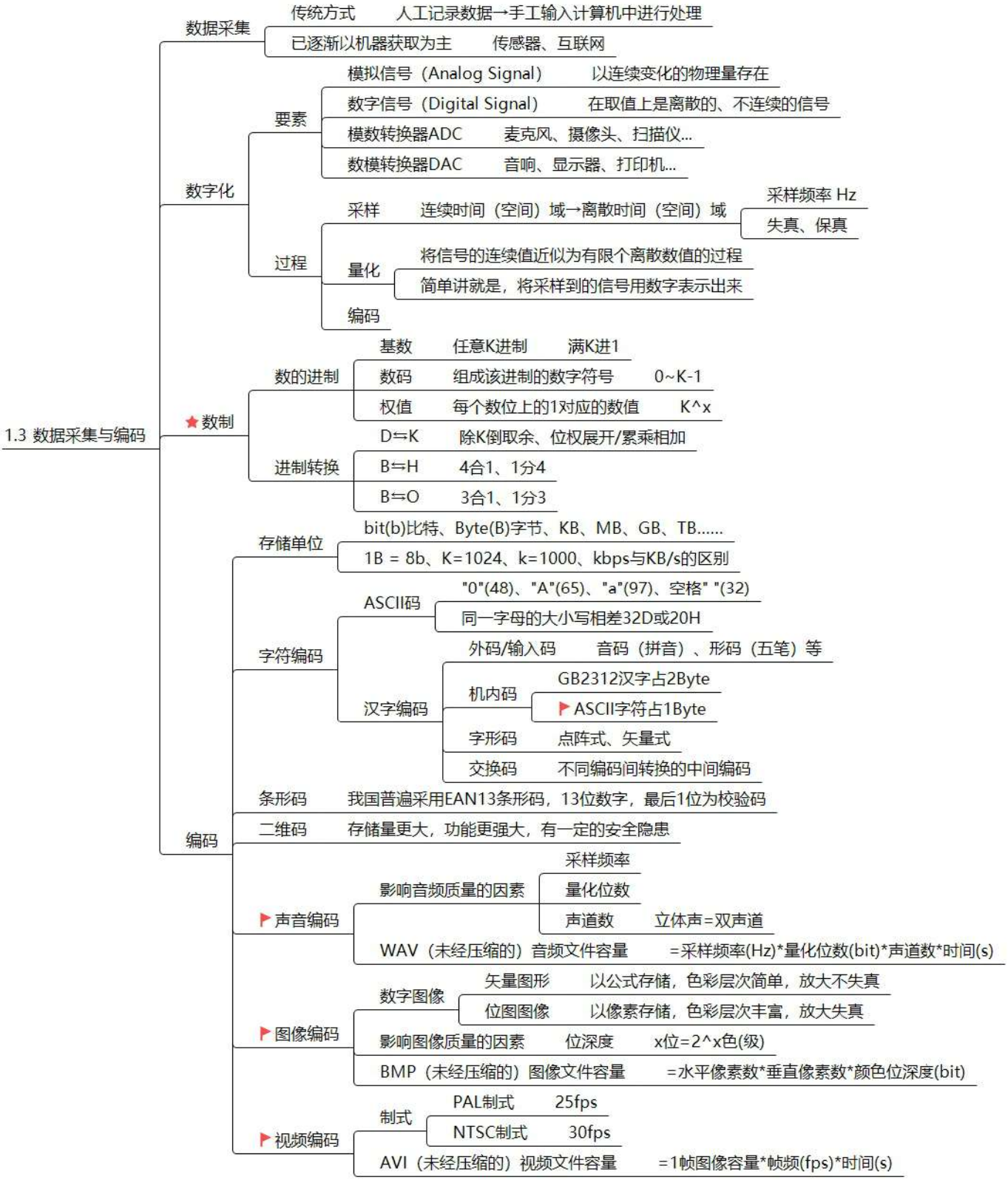
know-how

- 主要表现为收集、加工、应用、传播知识的能力，以及对事物发展的前瞻性看法，体现为一种卓越的判断力



关系

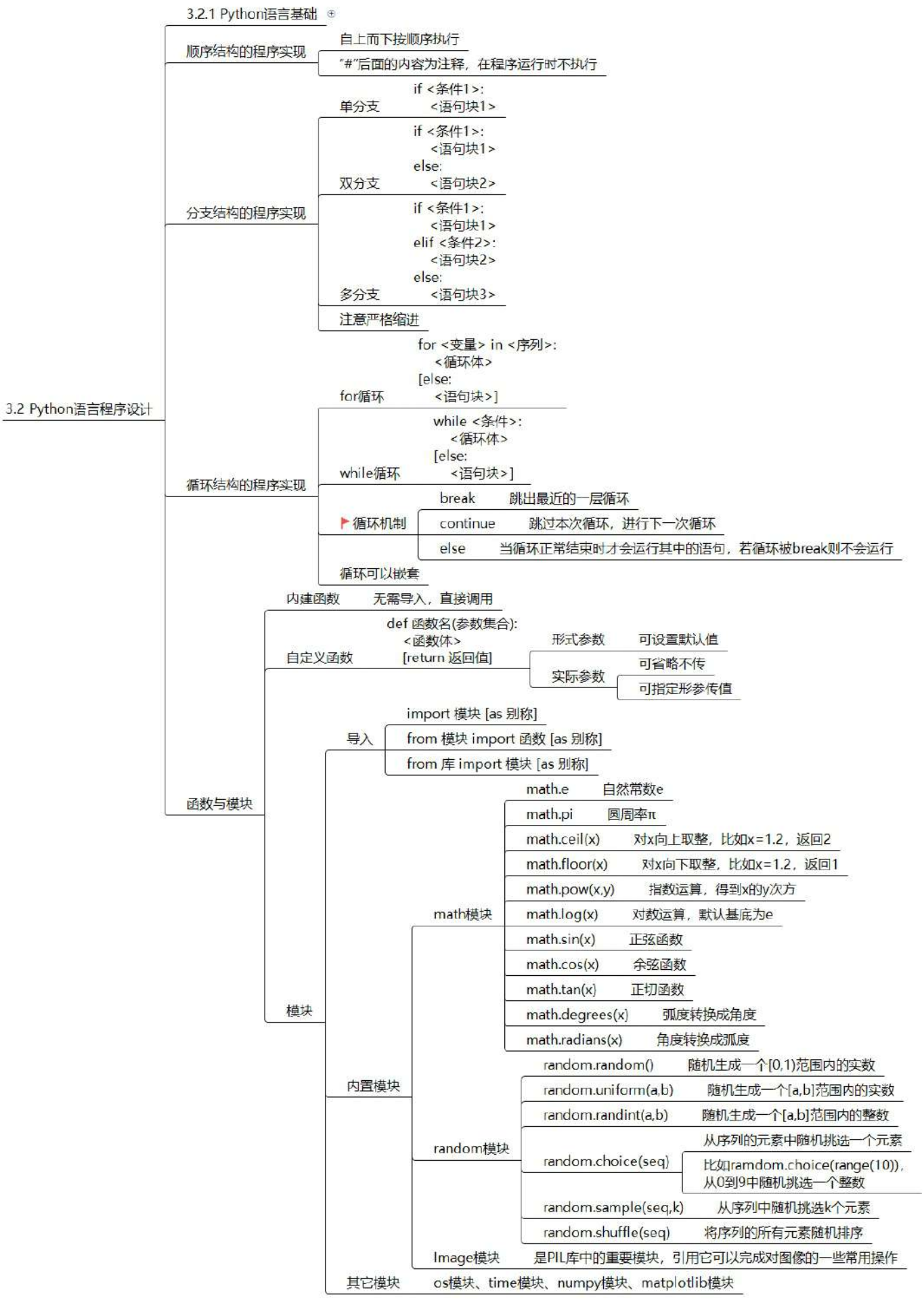
- 数据是信息的表示
- 信息是数据的内涵，是对数据的语义解释
- 知识是信息经过加工提炼后，形成的相应的抽象产物


1.3 数据采集与编码
数据采集
传统方式 人工记录数据→手工输入计算机中进行处理
已逐渐以机器获取为主 传感器、互联网
数字化
要素
模拟信号（Analog Signal） 以连续变化的物理量存在
数字信号（Digital Signal） 在取值上是离散的、不连续的信号
模数转换器ADC 麦克风、摄像头、扫描仪...
数模转换器DAC 音响、显示器、打印机...
过程
采样 连续时间（空间）域→离散时间（空间）域
采样频率 Hz
失真、保真
量化
将信号的连续值近似为有限个离散数值的过程
简单讲就是，将采样到的信号用数字表示出来
编码
★数制
数的进制
基数 任意K进制 满K进1
数码 组成该进制的数字符号 0~K-1
权值 每个数位上的1对应的数值 K^x
进制转换
D⇆K 除K倒取余、位权展开/累乘相加
B⇆H 4合1、1分4
B⇆O 3合1、1分3
编码
存储单位
bit(b)比特、Byte(B)字节、KB、MB、GB、TB.....
1B = 8b、K=1024、k=1000、kbps与KB/s的区别
字符编码
ASCII码
"0"(48)、"A"(65)、"a"(97)、空格" "(32)
同一字母的大小写相差32D或20H
汉字编码
外码/输入码 音码（拼音）、形码（五笔）等
机内码
GB2312汉字占2Byte
ASCII字符占1Byte
字形码 点阵式、矢量式
交换码 不同编码间转换的中间编码
条形码 我国普遍采用EAN13条形码，13位数字，最后1位为校验码
二维码 存储量更大，功能更强大，有一定的安全隐患
声音编码
影响音频质量的因素
采样频率
量化位数
声道数 立体声=双声道
WAV（未经压缩的）音频文件容量 =采样频率(Hz)*量化位数(bit)*声道数*时间(s)
图像编码
数字图像
矢量图形 以公式存储，色彩层次简单，放大不失真
位图图像 以像素存储，色彩层次丰富，放大失真
影响图像质量的因素 位深度 x位=2^x色(级)
BMP（未经压缩的）图像文件容量 =水平像素数*垂直像素数*颜色位深度(bit)
视频编码
制式
PAL制式 25fps
NTSC制式 30fps
AVI（未经压缩的）视频文件容量 =1帧图像容量*帧频(fps)*时间(s)


1.4 数据管理与安全
数据管理
人工管理
纸质档案
文件管理
文件类型（扩展名）
文本 txt （doc、xls、ppt）
图像 bmp、jpg、png、gif
音频 wav、mp3、wma
视频 avi、mp4、wmv
网页（超文本） htm、html
可执行文件 exe
文件夹+树形目录结构
数据库管理
传统数据库技术
结构化数据
数据独立性、数据可共享等特点
现代社会数据管理的主要方式
云计算、大数据
半结构化数据
非结构化数据
数据安全
威胁数据安全的因素
硬盘驱动器损坏、操作失误、黑客入侵、感染计算机病毒、遭受自然灾害
保护存储数据的介质（主动防护）
磁盘阵列、数据备份、异地容灾
提高数据本身的安全
数据加密（保密性）
明文、加密密钥、加密算法、密文
数据校验（完整性）
MD5、CRC、SHA-1
1.5 数据与大数据
概念
大数据代表着信息量大、速度快、种类繁多的信息资产，需要特定的技术和分析方法将其转化为价值
特征
数据体量巨大
数据量大并不一定就是大数据，用传统算法和数据库系统可以处理的海量数据不能算"大数据"
速度快
产生速度快
处理速度快
数据类型多
价值密度低
思维
抽样数据→全体数据
追求精确性→接受混杂性
强调因果关系→注重相关性
影响
让生活更便利
让决策更精准
带来新的就业需求
带来新的社会问题
信息泄露、数据安全、个人隐私、伦理道德

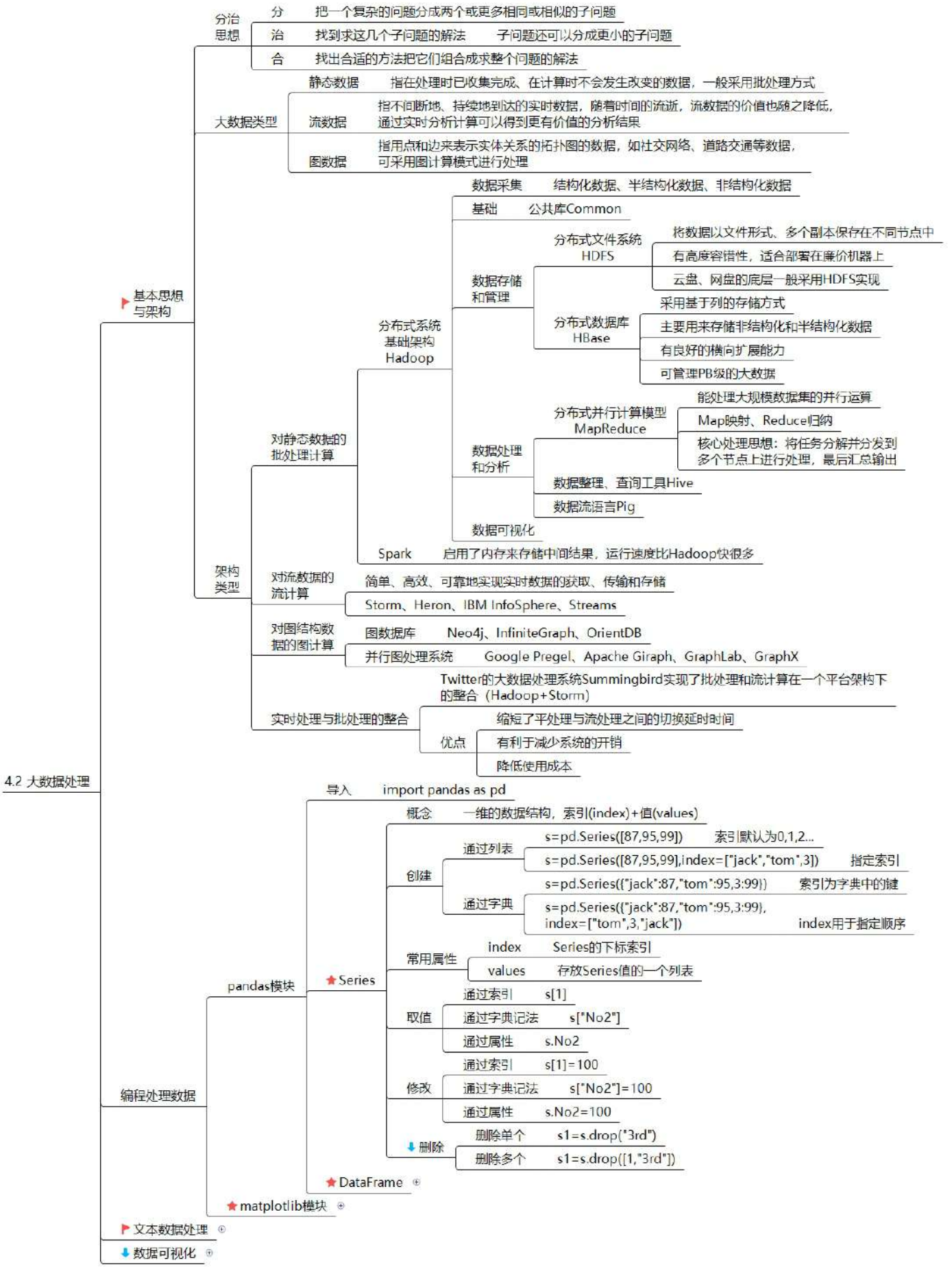

2.1 算法的概念及其描述
定义
广义：解决问题或完成任务的一系列步骤
狭义：用计算机解决问题的步骤，是为了解决问题而需要让计算机有序执行的、无歧义的、有限步骤的集合
特征
有穷性
可行性
确定性
0个或多个输入
1个或多个输出
要素
数据、运算、控制转移
描述
自然语言
通俗易懂，容易出现歧义，比较烦琐
流程图
结构清晰，寓意明确
伪代码
表示方法无统一、严格的规定，只要定义合理、表达正确即可
计算机语言
计算机可以理解并执行
发展历程
机器语言
汇编语言
高级语言
Python
2.2 算法的控制结构
顺序结构、分支结构、循环结构
2.3 用算法解决问题的过程
抽象与建模
提炼核心要素并加以确定或假设
用数学符号描述解决问题的计算模型
设计算法
设计
遵循算法的特征、围绕算法的要素
算法对数据的处理总体上需要经历三个步骤
输入数据
处理数据
输出处理结果
细化
"自顶向下，逐步细化"的结构化程序设计思想
描述算法
3.1 用计算机编程解决问题的一般过程
抽象与建模→设计算法→编写程序→调试运行程序

- 3.2.1 Python语言基础
  - 数据类型　整型、实型、字符串型、布尔型
  - 变量的命名规则
    - 字母、数字、下划线、（汉字）
    - 不能以数字开头
    - 字母区分大小写
    - 不能使用保留字
  - 赋值语句
    - 变量名=表达式
      - a=10
      - a,b=1,2　a,b=b,a
      - a=b=c=10
    - 复合赋值运算符　+=　-=　*=　/=　%=　//=　**=
  - 运算符
    - 算数运算符　**　优先于　*　/　//　%　优先于　+　-
    - 关系运算符　>　<　>=　<=　==　!=　in　not in
      - 字符串按"字典序"比大小
      - 范围比较 "a"<=ch<="z"
      - 特殊：字典中in用于判断"键是否存在"
    - 逻辑运算符　not　优先于　and　优先于　or
    - 运算符的优先级　小括号>算数运算>关系运算>逻辑运算
    - 字符串运算符　s="hello" + str(2) + "3" * 2
  - 表达式　变量、常量、运算符、圆括号
  - 基本数据结构
    - 字符串
      - 0个或多个字符组成的不可变序列
      - 操作
        - 创建　s="Hello"
        - 取值
          - 索引　从0开始，支持负数
          - 切片　s[起始：终止(不含)：方向和步长]
        - 不可增加、删除、修改
    - 列表
      - 0个或多个元素组成的可变序列
      - 操作
        - 创建　a=["He","llo"," ",666]
        - 取值
          - 索引
          - 切片
        - 增加
          - a.append(element)　在末尾添加element
          - a.insert(index,element)　在index处插入element
        - ▶删除
          - a.remove(element)　移除第一个element
          - a.pop()　移除最后一个元素
          - del a[start:end]　移除多个元素
        - 修改　a[index]=element　将x处的元素赋值为element
    - 字典
      - 一一对应的键值对
        - 键　字符串、数值
        - 值　任意类型
      - 操作
        - 创建　dict={key1:value1,key2:value2}
        - 取值
          - result=dict[key]
          - 元素是无序的，以键为索引，固不可切片
        - 增加　dict[new_key3]=new_value
        - ▶删除
          - dict.pop(key)
          - del dict[key]
        - 修改　dict[key1]=new_value
  - ★常见内建函数
    - print(x[,end=""])
    - input([prompt])　返回的一定是字符串型
    - int(x)　去掉小数部分，保留整数部分　▶注意区别：//是向下取整
    - float(x)
    - abs(x)
    - len(seq)
    - str(x)
    - chr(x)、ord(x)
    - round(x[,n])　四舍五入保留n位小数
    - max(s[,args...])、min(s[,args...])　参数可以是字符串、列表、字典
    - range(起始值,终值(不含),方向和步长)
      - 生成整数序列　range(1,10,2)
      - 步长值为1可省略　range(1,10)
      - 起始值为0可省略　range(10)


3.2.1 Python语言基础
3.2 Python语言程序设计
顺序结构的程序实现
自上而下按顺序执行
"#"后面的内容为注释，在程序运行时不执行
分支结构的程序实现
单分支
if <条件1>:
<语句块1>
双分支
if <条件1>:
<语句块1>
else:
<语句块2>
多分支
if <条件1>:
<语句块1>
elif <条件2>:
<语句块2>
else:
<语句块3>
注意严格缩进
循环结构的程序实现
for循环
for <变量> in <序列>:
<循环体>
[else:
<语句块>]
while循环
while <条件>:
<循环体>
[else:
<语句块>]
循环机制
break 跳出最近的一层循环
continue 跳过本次循环，进行下一次循环
else 当循环正常结束时才会运行其中的语句，若循环被break则不会运行
循环可以嵌套
函数与模块
内建函数 无需导入，直接调用
自定义函数
def 函数名(参数集合):
<函数体>
[return 返回值]
形式参数 可设置默认值
实际参数
可省略不传
可指定形参传值
模块
导入
import 模块 [as 别称]
from 模块 import 函数 [as 别称]
from 库 import 模块 [as 别称]
内置模块
math模块
math.e 自然常数e
math.pi 圆周率π
math.ceil(x) 对x向上取整，比如x=1.2，返回2
math.floor(x) 对x向下取整，比如x=1.2，返回1
math.pow(x,y) 指数运算，得到x的y次方
math.log(x) 对数运算，默认基底为e
math.sin(x) 正弦函数
math.cos(x) 余弦函数
math.tan(x) 正切函数
math.degrees(x) 弧度转换成角度
math.radians(x) 角度转换成弧度
random模块
random.random() 随机生成一个[0,1)范围内的实数
random.uniform(a,b) 随机生成一个[a,b]范围内的实数
random.randint(a,b) 随机生成一个[a,b]范围内的整数
random.choice(seq) 从序列的元素中随机挑选一个元素
比如ramdom.choice(range(10))，从0到9中随机挑选一个整数
random.sample(seq,k) 从序列中随机挑选k个元素
random.shuffle(seq) 将序列的所有元素随机排序
Image模块 是PIL库中的重要模块，引用它可以完成对图像的一些常用操作
其它模块 os模块、time模块、numpy模块、matplotlib模块

- 3.3 简单算法及其程序实现
  - 解析算法及其程序实现：根据问题的前提条件与所求结果之间的关系，找出求解问题的数学表达式，并通过表达式的计算来实现问题的求解
  - 枚举算法及其程序实现：把问题所有可能的解一一列举，然后判断每一个列举出的可能解是否为正确解
  - 文件操作
    - 打开：f=open("文件名.txt","r")
      - 模式
        - r：只读，从头开始
          - 文件不存在则会报错
        - w：只写，从头开始（覆盖原内容，原内容清空）
          - 文件不存在则会创建
        - a：只写，从尾开始（追加新内容，原内容保留）
          - 文件不存在则会创建
        - r+、w+、a+：可读可写
        - rb、rb+、wb、wb+、ab、ab+：以二进制格式打开
    - 读
      - f.read()：一次性读取文件的全部内容
      - f.read(size)：每次最多读取size个字节的内容
      - f.readlines()：每次读取一行内容，按行返回list
    - 写：f.write("内容")
    - 关闭：f.close()

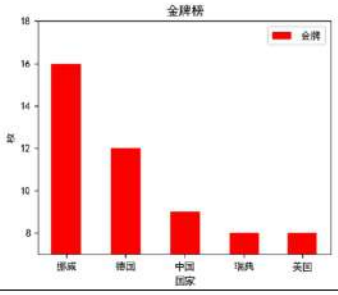
- 4.1 常用表格数据处理
  - 数据整理
    - 目的
      - 检测和修正错漏的数据
      - 整合数据资源
      - 规整数据格式
      - 提高数据质量
    - 常见问题及措施
      - 数据缺失：忽略含有缺失值的实例或属性；根据数据之间的关联性估计较准确的缺失值（平均值、中间值、概率统计值）
      - 数据重复：在进一步审核的基础上进行合并或删除
      - 数据异常：可能是要去掉的噪声，也可能是含有重要信息的数据对象
      - 逻辑错误：修正数据
      - 格式不一：数据转换
  - 数据计算
    - 数据处理
      - 日常简单的数据处理：EXCEL
      - 专业的数据处理和统计分析工具：SPSS、SAS、MATLAB
      - 计算机语言编程处理：R、Python、Java
    - EXCEL
      - 单元格：以行号和列号定位
      - 自动填充：有简单规律的数列、日期、公式
      - 公式
        - 常数
        - 运算符
          - 算数运算符：^ % * / + -
          - 比较运算符：> < >= <= <>
          - 文本连接运算符：&
        - 函数：SUM、AVERAGE、MAX、MIN；补充：RANK、COUNT
        - 单元格引用
          - 相对引用：A1
          - 绝对引用：A1
          - 连续引用：A2:D5
          - 不连续引用：A2:A5,D2:D5
          - 合并单元格的引用：最左上角的单元格
  - 数据图表呈现
    - 常见类型及特点
      - 柱形图/条形图：大小、多少
      - 折线图：变化趋势、起伏
      - 饼图：比例份额；只能呈现单系列
      - 雷达图：多项指标的对比分析
      - 散点图/气泡图：多个变量之间的关联情况
    - 数据区域的选择
      - 宽高对齐
      - 是否选择标题单元格
    - 图表的呈现
      - 受排序、筛选、直接或间接修改源数据的影响
      - 不受保留小数位数（四舍五入）的影响


4.2 大数据处理
基本思想与架构
分治思想
分 把一个复杂的问题分成两个或更多相同或相似的子问题
治 找到求这几个子问题的解法 子问题还可以分成更小的子问题
合 找出合适的方法把它们组合成求整个问题的解法
大数据类型
静态数据 指在处理时已收集完成、在计算时不会发生改变的数据，一般采用批处理方式
流数据 指不间断地、持续地到达的实时数据，随着时间的流逝，流数据的价值也随之降低，通过实时分析计算可以得到更有价值的分析结果
图数据 指用点和边来表示实体关系的拓扑图的数据，如社交网络、道路交通等数据，可采用图计算模式进行处理
架构类型
对静态数据的批处理计算
分布式系统基础架构Hadoop
数据采集 结构化数据、半结构化数据、非结构化数据
基础 公共库Common
数据存储和管理
分布式文件系统HDFS
将数据以文件形式、多个副本保存在不同节点中
有高度容错性，适合部署在廉价机器上
云盘、网盘的底层一般采用HDFS实现
分布式数据库HBase
采用基于列的存储方式
主要用来存储非结构化和半结构化数据
有良好的横向扩展能力
可管理PB级的大数据
数据处理和分析
分布式并行计算模型MapReduce
能处理大规模数据集的并行运算
Map映射、Reduce归纳
核心处理思想：将任务分解并分发到多个节点上进行处理，最后汇总输出
数据整理、查询工具Hive
数据流语言Pig
数据可视化
Spark 启用了内存来存储中间结果，运行速度比Hadoop快很多
对流数据的流计算
简单、高效、可靠地实现实时数据的获取、传输和存储
Storm、Heron、IBM InfoSphere、Streams
对图结构数据的图计算
图数据库 Neo4j、InfiniteGraph、OrientDB
并行图处理系统 Google Pregel、Apache Giraph、GraphLab、GraphX
实时处理与批处理的整合
Twitter的大数据处理系统Summingbird实现了批处理和流计算在一个平台架构下的整合（Hadoop+Storm）
优点
缩短了平处理与流处理之间的切换延时时间
有利于减少系统的开销
降低使用成本
编程处理数据
pandas模块
导入 import pandas as pd
Series
概念 一维的数据结构，索引(index)+值(values)
创建
通过列表
s=pd.Series([87,95,99]) 索引默认为0,1,2...
s=pd.Series([87,95,99],index=["jack","tom",3]) 指定索引
通过字典
s=pd.Series({"jack":87,"tom":95,3:99}) 索引为字典中的键
s=pd.Series({"jack":87,"tom":95,3:99}, index=["tom",3,"jack"]) index用于指定顺序
常用属性
index Series的下标索引
values 存放Series值的一个列表
取值
通过索引 s[1]
通过字典记法 s["No2"]
通过属性 s.No2
修改
通过索引 s[1]=100
通过字典记法 s["No2"]=100
通过属性 s.No2=100
删除
删除单个 s1=s.drop("3rd")
删除多个 s1=s.drop([1,"3rd"])
DataFrame
matplotlib模块
文本数据处理
数据可视化

- ★DataFrame
  - 概念：二维的数据结构，行索引(index)+列标题(columns)+值(values)，可以看作是共享同一个index的Series的集合
  - 创建
    - 通过字典：df=pd.DataFrame({"年龄":[16,17,17],"姓名":["jack","tom","lily"]}, columns=["姓名","年龄"],index=[1,"No2","3rd"])
      - columns用于指定列的顺序
      - index用于指定行索引
    - 通过列表：df=pd.DataFrame([["jack",16],["tom",17],["lily",17]], columns=["姓名","年龄"],index=[1,"No2","3rd"])
      - columns用于指定列标题
      - index用于指定行索引
    - 二维数据文件
      - 读取
        - df=pd.DataFrame.read_excel("test.xlsx")
        - df=pd.DataFrame.read_csv("test.csv")
      - 写入
        - pd.DataFrame.to_excel("output.xlsx")
        - pd.DataFrame.to_csv("output.csv")
  - 常用属性
    - index　行索引
    - columns　存放各列的列标题
    - values　存放值的二维列表
    - T　行列转置（不改变原数据对象）
  - 取值
    - 列
      - 通过字典记法：单列　df["姓名"]；⬇多列　df[["姓名","年龄"]]
      - 通过属性　df.姓名
    - 行
      - 通过切片　df[0:2]
      - ★通过布尔型数据选取满足条件的行
        - df1=df[df["年龄"]>16]
        - df1=df[(df["年龄"]>16) & (df["性别"]=="男")]
        - df1=df[(df["年龄"]>16) | (df["性别"]=="男")]
        - df1=df[~(df["年龄"]>16)]
    - ★单个值
      - 通过at方法（index+columns）　df.at[1,"姓名"]
      - 通过字典（columns+index）　df["姓名"][1]
      - 通过属性（columns+index）　df.姓名[1]
      - 通过values属性（行号+列号）　df.values[1][0]　特别注意values方式与其它3种方式的区别
  - 修改
    - 列
      - 通过字典记法　df["年龄"]=[16,16,17]
      - 通过属性　df.年龄=[16,16,17]
    - 行　通过切片
      - df[0:1]=["Jack",17]
      - df[0:2]=[["Jack",17],["Tom",18]]
    - ★单个值　通过at方法（index+columns）　df.at["No2","姓名"]="tom2"
  - 函数
    - 基本统计值　count()、sum()、mean()、max()、min()、describe()　df["年龄"].max() 或 df.年龄.max()
    - 选取数据记录
      - head()　df1=df.head(3)　返回前n项，默认5项
      - tail()　df1=df.tail(3)　返回后n项，默认5项
    - ★分组
      - groupby()　df1=df.groupby("年龄",as_index=False)
        - as_index：是否以组标签作为新的索引，默认为True
      - groupby()后一般接一个基本统计函数　df1=df.groupby("年龄",as_index=False).count()
    - ★排序
      - 按值排 sort_values()　df1=df.sort_values("年龄",ascending=False)
      - ⬇按索引排 sort_index()　df1=df.sort_index(ascending=False)
      - 排序参数
        - ascending　True：升（默认）、False：降
        - axis　0：以行为整体，纵向排序（默认）、1：以列为整体，横向排序
    - ⬇增加
      - 增加行　append()　df1=df.append({字典},ignore_index=True)
        - ignore_index：是否重新安排index以避免出现相同的index，默认为False
      - 插入列　insert()　df.insert(新列标题的位置索引值,新列标题名,[新值的列表])
        - 特别注意：insert方法会改变原数据对象df，无返回值；另外，参数inplace=True也会让一般的函数改变原数据对象
    - 删除　drop()
      - 删除行　df1=df.drop(1) 或 df1=df.drop("No2")
      - 删除列　df1=df.drop("年龄",axis=1)
    - ⬇合并　concat()　df4=pd.concat([df1,df2,df3],ignore_index=True)
    - 修改行名、列名　rename()
      - 修改行索引　df.rename(index={1:"1st","No2":2})
      - 修改列标题　df.rename(columns={"姓名":"name","年龄":"age"})
    - ★默认参数
      - as_index　默认为True,以组标签作为新的索引
      - ascending　默认为True，升序
      - axis　默认为0，以行为整体，纵向排序
      - ignore_index　默认为False，不重新安排index
      - inplace　默认为False，不会更新原数据对象

- 4.2 大数据处理
  - 基本思想与架构
  - 编程处理数据
    - pandas模块
    - ★matplotlib模块
      - 导入　import matplotlib.pyplot as plt
      - 创建一个图表对象　figure()　plt.figure(figsize=(8,4))
      - 绘制图表
        - 绘制线形图　plot()　plt.plot([x轴数据],[数值],label="标签名",color="r",linewidth=2)
        - 绘制垂直柱形图　bar()　plt.bar([x轴数据],[数值],color="r")　图表中水平轴是x轴，竖直轴是y轴
        - 绘制水平柱形图　barh()　plt.barh([x轴数据],[数值],color="r")　图表中竖直轴是x轴，水平轴是y轴；数据排布方向：从原点到远离原点
        - 绘制饼图　pie()　plt.pie([数值],labels=[标签])
        - 绘制散点图　scatter()　plt.scatter([x轴数据],[数值],label="标签名",color="r")
      - 设置其它参数
        - 设置图表的标题　title()　plt.title("我是标题")
        - 设置x、y轴的取值范围　xlim()、ylim()　plt.xlim(0,10)
        - 设置x、y轴的标签　xlabel()、ylabel()　plt.xlable("我是x轴标题")
        - 显示图例　legend()　plt.legend()
      - 显示创建的所有绘图对象　show()　plt.show()
      - 实例

```
import pandas as pd
import matplotlib.pyplot as plt
df=pd.read_excel("medal.xlsx")
df=df.sort_values("金牌",ascending=False).head()
print(df)
plt.rcParams["font.sans-serif"]=["SimHei"]#字体
plt.figure(figsize=(5,4))
plt.bar(df["国家"],df["金牌"],label="金牌",color="r")
plt.title("金牌榜")
plt.ylim(7,18)
plt.xlabel("国家")
plt.ylabel("枚")
plt.legend()
plt.show()
```



  - 文本数据处理
    - 目的　从大规模的文本数据中提取出符合需要的、感兴趣的和隐藏的信息
    - 应用　搜索引擎、情报分析、自动摘要、自动校对、论文查重、文本分类、垃圾邮件过滤、机器翻译、自动应答等方面
    - 文本数据处理的一般过程
      - 文本数据源　文本内容是非结构化的数据
      - (中文)分词
        - 基于词典(字符匹配)　分析句子时与词典中的词语进行对比
          - jieba模块
            - cut_all=False　精确模式(默认模式)　"武松","打死老虎"
            - cut_all=True　全模式　"武松","打死","打死老虎","死老虎","老虎"
        - 基于统计　依据上下文相邻字出现的频率统计；一般与"基于词典"结合使用
        - 基于规则　让计算机模拟人的理解方式，根据大量的现有资料和规则进行学习；目前还处于实验阶段
      - 特征提取　目前大多采用词作为文本的特征项，称作特征词，而不采用字和短语
        - 减少特征词的数量的方式　根据专家的知识挑选有价值的特征；用数学建模的方法构造评估函数自动选取特征
        - 未来用深度学习、大数据分析，特征提取将更加准确科学
      - 数据分析(与应用)
        - 标签云
          - 概念　将文本中复杂的或者难以通过文字表达的内容和规律以视觉符号的形式表达出来
          - 依据　用词频表现文本特征，将关键词按照一定的顺序和规律排列
          - 应用　报纸、杂志
          - WordCloud模块
        - 文本情感分析
          - 概念　通过计算机技术对文本的主观性、观点、情绪、极性进行挖掘和分析，对文本的情感倾向作出分类判断
          - 依据　根据分析的粒度不同，分为词语级、语句级、整篇文章级三类
          - 应用　网络舆情监控、用户评论分析与决策、信息预测
      - 结果呈现　数据可视化
  - 数据可视化
    - 概念　将数据以图形图像等形式表示，直接呈现数据中蕴含信息的处理过程
    - 作用
      - 快捷观察与追踪数据
      - 实时分析数据
      - 增强数据的解释力与吸引力
    - 基本方法
      - 有关时间趋势的可视化　柱形图、折线图
      - 有关比例的可视化　饼图、环形图
      - 有关关系的可视化　散点图、气泡图
      - 有关差异的可视化　雷达图
      - 有关空间关系的可视化　地理数据图
    - 工具
      - 软件实现数据可视化　大数据魔镜、Gephi、Tableau
      - 编程实现数据可视化　Python、R
      - 可视化工具库
        - 基于JavaScript的D3.js、Highcharts、Google Charts
        - 基于Python的matplotlib

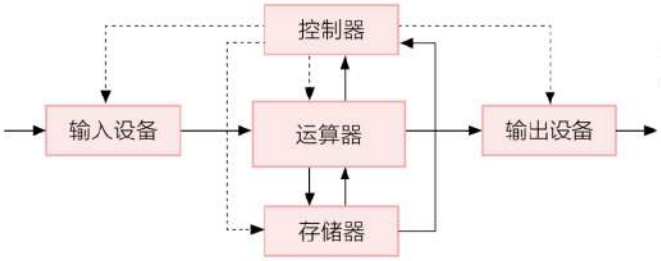
- 4.3 ⬇ 大数据典型应用
  - 数据成为核心资产，数据规模以及运用数据的能力成为各行业发展的推动力
  - 大数据时代，数据使用的关键是"数据再利用"
  - 智能交通　交通信息服务系统、交通管理系统、公共交通系统、车辆控制系统、货运管理系统、电子收费系统、紧急救援系统
  - 电子商务　精准营销、供应链管理、智能网站
- 5.1 ★ 人工智能的产生与发展
  - 发展历程
    - 概念
      - 以机器（计算机）为载体，模仿、延伸、扩展人类智能，其与人类或其他动物所呈现的生物智能有重要区别
      - 三种主要方法　符号主义、联结主义、行为主义
    - 从计算到智能测试
      - 三种等效的计算机制
        - 原始递归函数
        - lambda演算
        - 图灵机
          - 现代计算机理论之父：图灵
          - 可计算任务=图灵可停机任务
      - 图灵测试　测试机器是否具备智能的一种方法
    - 人工智能登上历史舞台　人工智能达特茅斯夏季研讨会上，首次提出"人工智能"概念
    - 符号主义：以符号主义表达和推理为代表的人工智能（基于推理）
      - 或称为　逻辑主义、心理学派、计算机学派
      - 原理　对符号的推理和运算
      - 包含　手工构造知识库+推理引擎
      - 知识工程　对人类知识进行逻辑编码，然后通过推理引擎对编码知识进行操作，形成某一领域的"专家系统"
        - 沃森（问答）
        - NELL（语言）
        - DENDRAL（化学）
        - MYCIN（医学）
        - 深蓝（象棋）
      - 特征　解释性强、可拓展性弱
      - 瓶颈问题　知识的精确化编码阻碍了符号人工智能的发展
    - 联结主义：数据驱动的人工智能方法（基于数据）
      - 或称为　仿生学派、生理学派
      - 原理　模仿人类大脑中神经元来进行认知推理
      - 深度学习　一种对原始数据所蕴含的特征模式进行学习的算法模型
      - 应用
        - 自然语言处理、知识图谱构建、图像分类、语音识别、视频运动提取
        - 通过"深层残差卷积网络"进行图像分类
        - AlphaGo(Lee/Master) 战胜人类棋手（结合了强化学习）
    - 行为主义：问题引导下的人工智能学习方法（基于反馈）
      - 或称为　进化主义、控制论学派
      - 原理　通过"交互-反馈"，在与环境的交互中不断学习，从而提升自己的智能水平
      - 试错学习（强化学习）　事先不知道最终答案，不断尝试各种解决问题的可能途径，根据结果反馈来调整相应的学习方法
      - 应用
        - AlphaGo Zero 100:0 AlphaGo Lee
        - 德州扑克人工智能Libratus
- 5.2 ★ 人工智能的应用
  - 领域人工智能　依赖于领域知识和数据的人工智能　深蓝、沃森
  - 跨领域人工智能　智能系统从一个领域快速跨越到另外一个领域
    - AlphaGo（围棋→电力控制）
    - 沃森（游戏→医疗）
  - 混合增强智能
    - 多种智能体的混合形式，将人的作用或人的认知模型引入人工智能系统
      - 达芬奇外科手术机器人
      - 人机协同的太空机器人
    - 人类智能是智能回路的总开关
- 5.3 ⬇ 人工智能对社会的影响
  - 改善人类生活
    - 智能家居　智能冰箱、智能温控、智能安防、智能扫地机器人
    - 智慧城市　城市大脑
    - 智能出行　自动驾驶
    - 智能购物　无人超市、无人酒店、无人餐厅
  - 促进经济发展
    - 创造了一种虚拟的劳动力
    - 提高生产力、创造全新的产品和服务
    - 推动人工智能和实体经济结合
  - 带来社会担忧
    - 取代工作岗位
    - 可能威胁人类安全

必修二《信息系统与社会》

- 1.1 信息技术与信息系统
  - 信息技术
    - 定义：是指获取、传输、存储、加工和表达信息的各种技术的总和
    - 主要包括：计算机技术、计算机网络技术、通信技术（电视、电话等）
    - 发展历史
      - 前机械时期 1450以前：符号、文字、造纸术
      - 机械时期 1450~1840：活字印刷、计算尺、加法器、莱布尼茨计算器
      - 电子机械时期 1840~1940：方式：电子脉冲；电报、电话、收音机、电子机械计算机
      - 电子化时期 1940~至今：首台电子管通用计算器ENIAC、PC；四个阶段：电子管→晶体管→集成电路→大规模超大规模集成电路
    - 促进社会进步：以计算机为核心 → 以互联网为核心 → 以数据为核心 → 新的发展形态；数据已成为重要的生产资料
  - 信息系统
    - 定义：是指由硬件软件设施、通信网络、数据和用户构成的人机交互系统
    - 类型：业务处理系统、决策支持系统、知识管理系统、学习管理系统、数据库管理系统、办公信息系统等
- 1.2 信息系统的组成与功能
  - ★组成：硬件、软件、数据、通信网络、用户
  - 功能：数据收集和输入、数据存储、数据传输、数据加工处理、数据输出、数据查询
- 1.3 信息系统的应用
  - 应用分类
    - 按规模：简单系统、复杂系统
    - 按发展阶段：数据（事务）处理系统→管理信息系统→决策信息系统
    - 按应用领域：通用、制造业、医疗保健、学校管理、银行等
  - 优势
    - 规范工作流程，提高工作效率
    - 跨越时空限制，服务随时随处
    - 基于数据分析，支持科学决策
    - 便捷保存数据，利于共享追踪
  - 局限性
    - 对外部环境有依赖性
    - 本身有安全隐患
    - 技术门槛可能加剧数字鸿沟
- 1.4 信息社会及其发展
  - 基本内涵
    - 是以人为本的：社会发展必须以人为中心；只有以人为本，才能体现信息社会的包容性；以人为本的社会才能实现全面发展
    - 是可持续发展的：既能满足当代人的需要，又不对后代人的生存能力构成危害
    - 是以信息和知识作为重要资源的
  - 主要特征（从四个维度阐述）
    - 信息经济：是信息社会最基本的经济形态，也是决定信息社会发展水平高低的重要因素；以创新为主要驱动力
    - 网络社会：信息服务的可获得性：高速、泛在、好用、便宜；社会发展的全面性：基本衣食住行→健康生活和人与自然和谐发展
    - 在线政府：科学决策、公开透明、高效治理、互动参与
    - 数字生活：生活工具数字化、生活方式数字化、生活内容数字化
  - 信息社会的发展过程
    - 信息社会指数ISI：信息经济指数、网络社会指数、在线政府指数、数字生活指数；以不同权重计算而得出
    - 划分阶段
      - 准备阶段：起步期（ISI < 0.3）：加快基础建设；转型期（0.3≤ISI < 0.6）：加快调整与改革，加强教育
      - 发展阶段：初级阶段（0.6≤ISI < 0.8）：改进机制体制；中级阶段（0.8≤ISI < 0.9）：关注弱势；高级阶段（0.9≤ISI）：鼓励创新

2.1 计算机硬件

- 概念：指信息系统中看得见、摸得着的设备，包含计算机硬件、移动终端硬件、通信网络设备
- 计算机的发展历程
  - 电子管→晶体管→集成电路（中小规模）→大规模超大规模集成电路
  - 未来向巨型化、微型化、智能化、网络化等方向发展
- ★ 计算机硬件组成（冯·诺依曼体系结构）
  - 运算器
  - 控制器
  - CPU
    - 性能指标：主频、字长、核心数量、高速缓存
    - 额外扩展：FPU、Cache、存储管理部件、GPU
    - 延伸：云计算
  - 存储器
    - 功能：存放程序（指令）和数据（操作数、运算结果）
    - 按用途分类
      - 主存储器（内存）
        - ROM只读存储器：断电信息保留
        - RAM随机存取存储器：断电信息丢失
      - 辅助存储器（外存）
        - 硬盘
          - 机械硬盘HDD
          - 固态硬盘SSD
        - 闪存盘：U盘
        - 延伸：云存储
      - 断电信息保留
      - 高速缓冲存储器Cache：读写速度：CPU > Cache > 内存 > 外存
    - 性能指标：存储容量、读写速度
  - 输入设备：键盘、鼠标、摄像头、麦克风、扫描仪、传感器等
  - 输出设备：显示器、打印机、音箱、耳机、控制器等
- ▶ 计算机的工作原理

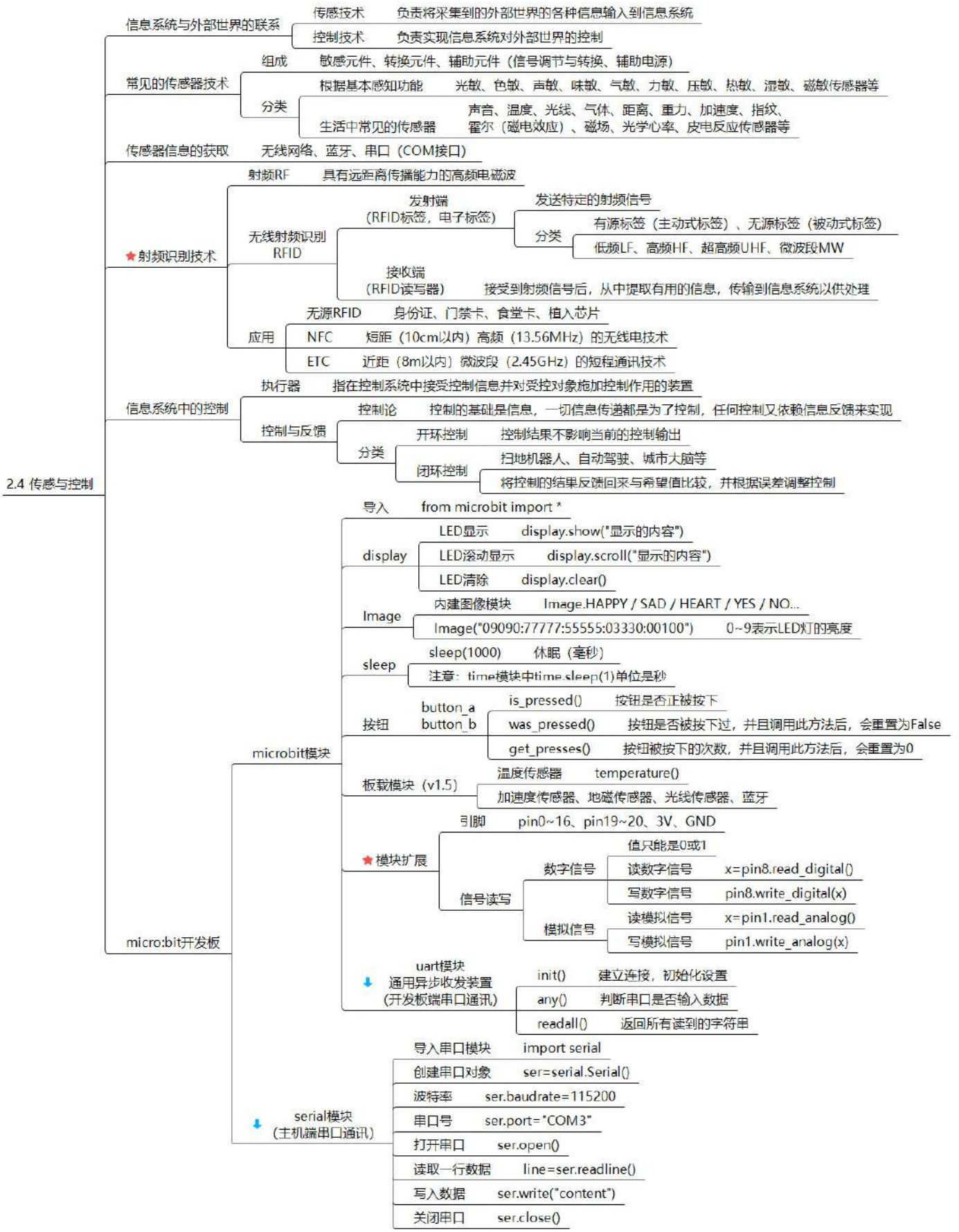


2.2 计算机软件

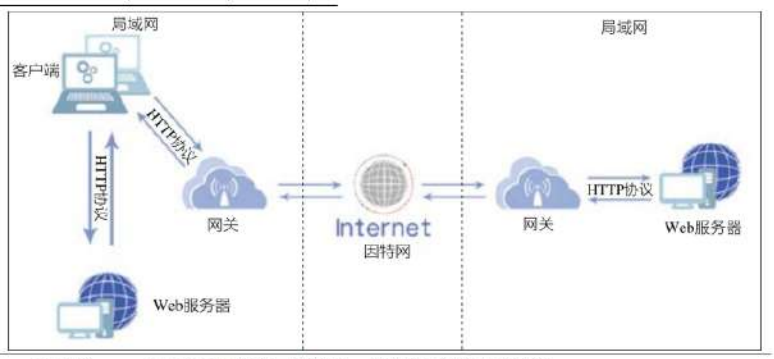
- 概念：是指在计算机上运行的程序及其数据和文档的总和
- 系统软件
  - 主要功能：对计算机系统的全部软硬件和数据资源进行统一控制、调度和管理
  - 分类
    - 操作系统（最重要）：Windows、Mac OS、Linux、Unix
    - 计算机语言处理程序：编译程序、Python解释器、汇编语言汇编器
    - 数据库管理系统：Oracle、MySQL
    - 其它辅助程序软件
- 应用软件
  - 概念：利用计算机的软、硬件资源，为解决某一应用领域的某个实际问题而专门开发的软件
  - 常见软件
    - 办公软件：Microsoft Office(Word、Excel、PowerPoint)、金山WPS
    - 工具软件：系统维护、系统美化、图像编辑、音视频处理等
    - 系统自带的应用软件：计算器、画图、记事本等
    - 如果没有现成的能解决问题的应用软件，可以自行定制开发：图书管理系统

2.3 移动终端

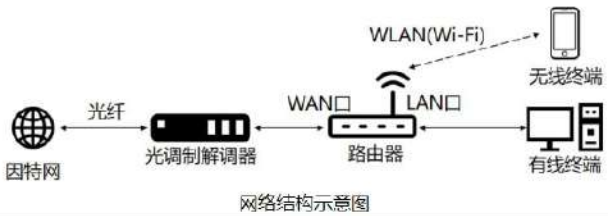
- 硬件
  - 采用计算机经典体系结构
  - 保持合理平衡
    - 较小的尺寸
    - 较低的功耗
    - 较高的性能
- 软件
  - 系统软件：Android（基于Linux）、iOS、Windows
  - 应用软件
- ▶ 工作原理
  - 与计算机相同
  - 移动性：基本没有空间上的限制
  - 智能性：主要体现在传感器的植入
- 手机的性能指标：CPU、存储器（RAM、ROM）、屏幕


2.4 传感与控制
信息系统与外部世界的联系
传感技术 负责将采集到的外部世界的各种信息输入到信息系统
控制技术 负责实现信息系统对外部世界的控制
常见的传感器技术
组成 敏感元件、转换元件、辅助元件（信号调节与转换、辅助电源）
分类
根据基本感知功能 光敏、色敏、声敏、味敏、气敏、力敏、压敏、热敏、湿敏、磁敏传感器等
生活中常见的传感器 声音、温度、光线、气体、距离、重力、加速度、指纹、霍尔（磁电效应）、磁场、光学心率、皮电反应传感器等
传感器信息的获取 无线网络、蓝牙、串口（COM接口）
★射频识别技术
射频RF 具有远距离传播能力的高频电磁波
无线射频识别RFID
发射端（RFID标签，电子标签） 发送特定的射频信号
分类 有源标签（主动式标签）、无源标签（被动式标签）
低频LF、高频HF、超高频UHF、微波段MW
接收端（RFID读写器） 接受到射频信号后，从中提取有用的信息，传输到信息系统以供处理
应用
无源RFID 身份证、门禁卡、食堂卡、植入芯片
NFC 短距（10cm以内）高频（13.56MHz）的无线电技术
ETC 近距（8m以内）微波段（2.45GHz）的短程通讯技术
信息系统中的控制
执行器 指在控制系统中接受控制信息并对受控对象施加控制作用的装置
控制与反馈
控制论 控制的基础是信息，一切信息传递都是为了控制，任何控制又依赖信息反馈来实现
分类
开环控制 控制结果不影响当前的控制输出
闭环控制 扫地机器人、自动驾驶、城市大脑等
将控制的结果反馈回来与希望值比较，并根据误差调整控制
micro:bit开发板
microbit模块
导入 from microbit import *
display
LED显示 display.show("显示的内容")
LED滚动显示 display.scroll("显示的内容")
LED清除 display.clear()
Image
内建图像模块 Image.HAPPY / SAD / HEART / YES / NO...
Image("09090:77777:55555:03330:00100") 0~9表示LED灯的亮度
sleep
sleep(1000) 休眠（毫秒）
注意：time模块中time.sleep(1)单位是秒
按钮
button_a
button_b
is_pressed() 按钮是否正被按下
was_pressed() 按钮是否被按下过，并且调用此方法后，会重置为False
get_presses() 按钮被按下的次数，并且调用此方法后，会重置为0
板载模块（v1.5）
温度传感器 temperature()
加速度传感器、地磁传感器、光线传感器、蓝牙
★模块扩展
引脚 pin0~16、pin19~20、3V、GND
信号读写
数字信号 值只能是0或1
读数字信号 x=pin8.read_digital()
写数字信号 pin8.write_digital(x)
模拟信号
读模拟信号 x=pin1.read_analog()
写模拟信号 pin1.write_analog(x)
uart模块
通用异步收发装置
（开发板端串口通讯）
init() 建立连接，初始化设置
any() 判断串口是否输入数据
readall() 返回所有读到的字符串
serial模块
（主机端串口通讯）
导入串口模块 import serial
创建串口对象 ser=serial.Serial()
波特率 ser.baudrate=115200
串口号 ser.port="COM3"
打开串口 ser.open()
读取一行数据 line=ser.readline()
写入数据 ser.write("content")
关闭串口 ser.close()

- 2.5 网络系统
  - 组成与构建
    - 功能
      - 数据通信　是通信技术和计算机技术相结合而产生的一种通信方式，是网络系统最基本的功能
      - 资源共享　是指网络中所有的软件、硬件、数据资源等能被网络中的所有用户共同使用
      - 分布处理　分布式处理是指将不同地点或具有不同功能或拥有不同数据的多台计算机通过通信网络连接起来，并在控制系统的统一管理控制下，协调地完成大规模信息处理任务
    - ★分类
      - 计算机网络　局域网LAN、城域网MAN、广域网WAN
      - 移动通信网络　1~5G 五代移动通信技术
      - 广播电视网络（混合光纤同轴网络）　频带宽、容量大、功能多、成本低、抗干扰能力强、支持多种业务
    - 典例：网页传输过程
      - URL，资源定位符　网络协议 + 服务器地址 + 文件名
      - HTTP，超文本传输协议
      - HTML，超文本标记语言（网页文件）

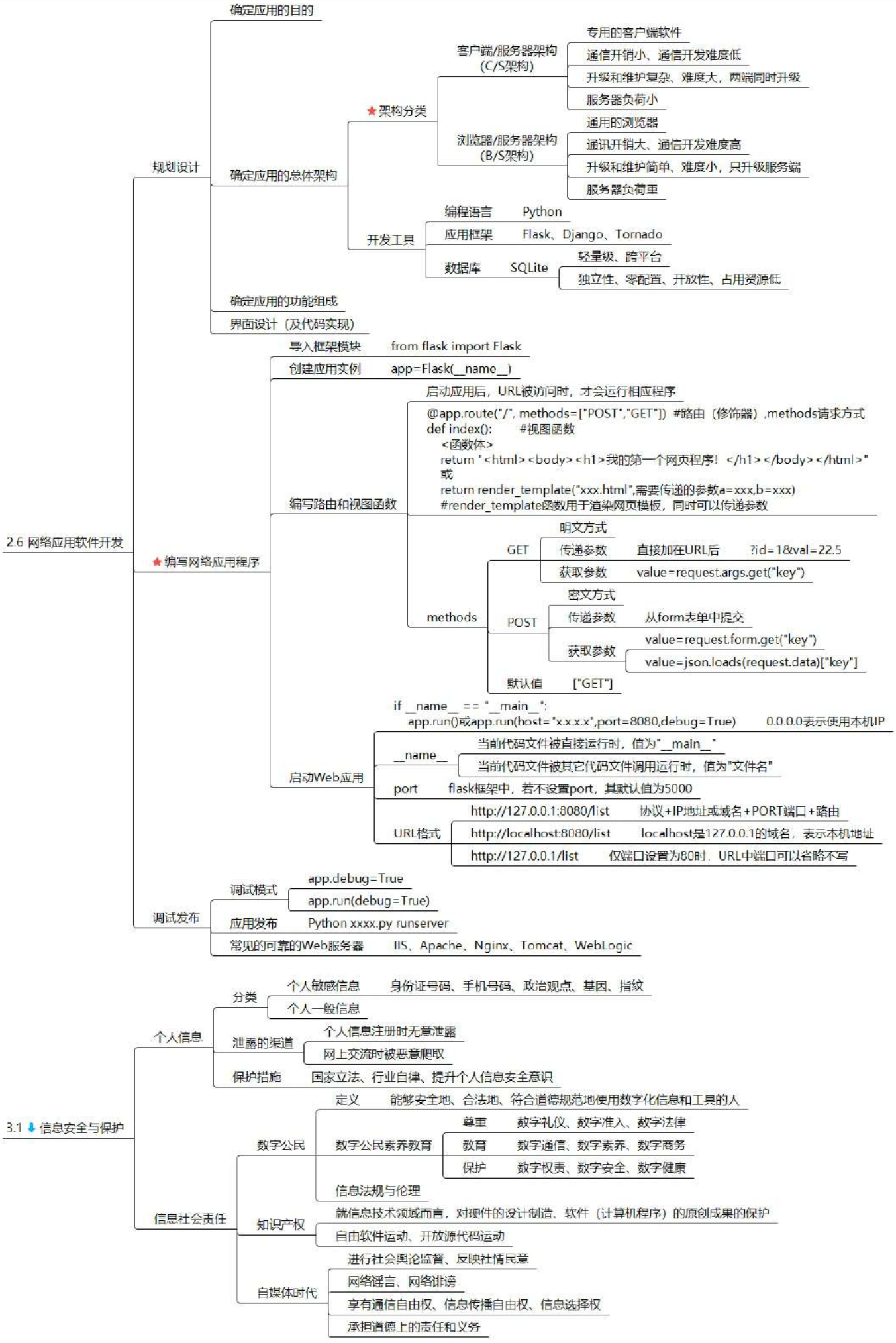


    - ▶组成
      - 计算机系统
        - 服务器　负责数据处理和网络控制，并构成网络的主要资源
        - 终端　是网络中数量大、分布广的设备，是用户进行网络操作、实现人机对话的工具
      - 数据通信系统
        - 传输介质
          - 有线　光纤、电缆、双绞线等
          - 无线　无线电微波信号（Wi-Fi、蓝牙、RF）、红外信号等
        - 网络互联设备　调制解调器、路由器、交换机
      - 网络软件和网络协议
        - 网络软件
          - 作用
            - 授权给用户，使用户能访问网络资源，并且方便、安全地使用网络
            - 管理和调度网络资源，提供网络通信和用户所需的各种网络服务
          - 包括
            - 网络操作系统NOS
              - 资源管理、网络通信、任务调度、提供接口
              - 常见的：UNIX、Windows Server、Linux
            - 通信软件
            - 管理和服务软件
        - 网络协议
          - 实现网络不同终端、不同网络之间相互识别和正确通信的一组标准和规则
          - ▶三层协议
            - AP，应用程序协议，顶层　将传输的信息转换成人类能识别的内容
              - 如HTTP、FTP、SMTP、POP3等
            - TCP，传输控制协议，中层　管理被传送内容的完整性
            - IP，网际协议，底层　将信息从一个地方传送到另一个地方
          - 注意：TCP/IP协议是其它所有网络协议（HTTP、FTP、Telnet等）的基础协议
    - 构建
      - (1)确定网络结构，连接网络设备
        - 光调制解调器
        - 路由器　WAN口、LAN口、WLAN（Wi-Fi）

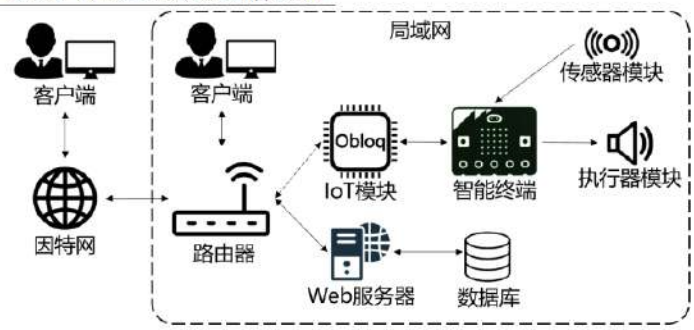


网络结构示意图

      - (2)互联网的接入
        - 设置路由器接入因特网的方式
          - PPPoE（拨号上网）
            - 适用于个人用户
            - 输入ISP提供的用户名和密码
          - 动态地址
          - 静态地址
        - 启用路由器的网络地址转换功能（默认自动开启）
      - (3)有线终端设备接入局域网
        - 终端设备的IP需要和路由器IP处于相同网段
        - 方式
          - 动态地址
            - 上级路由器开启DHCP服务，该服务会自动给接入的设备分配IP地址、子网掩码、默认网关
            - PC机、机顶盒一般会选用
          - 静态地址
            - 手动配置IP地址、子网掩码、默认网关
            - 路由器、交换机、服务器等一般会选用
      - (4)移动设备接入局域网　SSID（WLAN名称）、PASSWORD、加密方式、无线信道（2.4GHz、5GHz）
      - (5)网络连通性测试　浏览终端网页是否正常


2.6 网络应用软件开发
规划设计
确定应用的目的
确定应用的总体架构
★架构分类
客户端/服务器架构(C/S架构)
专用的客户端软件
通信开销小、通信开发难度低
升级和维护复杂、难度大，两端同时升级
服务器负荷小
浏览器/服务器架构(B/S架构)
通用的浏览器
通讯开销大、通信开发难度高
升级和维护简单、难度小，只升级服务端
服务器负荷重
开发工具
编程语言 Python
应用框架 Flask、Django、Tornado
数据库 SQLite
轻量级、跨平台
独立性、零配置、开放性、占用资源低
确定应用的功能组成
界面设计（及代码实现）
★编写网络应用程序
导入框架模块 from flask import Flask
创建应用实例 app=Flask(__name__)
编写路由和视图函数
启动应用后，URL被访问时，才会运行相应程序
@app.route("/", methods=["POST","GET"]) #路由（修饰器）,methods请求方式
def index(): #视图函数
<函数体>
return "<html><body><h1>我的第一个网页程序！</h1></body></html>"
或
return render_template("xxx.html",需要传递的参数a=xxx,b=xxx)
#render_template函数用于渲染网页模板，同时可以传递参数
methods
GET
明文方式
传递参数 直接加在URL后 ?id=1&val=22.5
获取参数 value=request.args.get("key")
POST
密文方式
传递参数 从form表单中提交
获取参数
value=request.form.get("key")
value=json.loads(request.data)["key"]
默认值 ["GET"]
启动Web应用
if __name__ == "__main__":
app.run()或app.run(host="x.x.x.x",port=8080,debug=True) 0.0.0.0表示使用本机IP
__name__
当前代码文件被直接运行时，值为"__main__"
当前代码文件被其它代码文件调用运行时，值为"文件名"
port flask框架中，若不设置port，其默认值为5000
URL格式
http://127.0.0.1:8080/list 协议+IP地址或域名+PORT端口+路由
http://localhost:8080/list localhost是127.0.0.1的域名，表示本机地址
http://127.0.0.1/list 仅端口设置为80时，URL中端口可以省略不写
调试发布
调试模式
app.debug=True
app.run(debug=True)
应用发布 Python xxxx.py runserver
常见的可靠的Web服务器 IIS、Apache、Nginx、Tomcat、WebLogic
3.1 信息安全与保护
个人信息
分类
个人敏感信息 身份证号码、手机号码、政治观点、基因、指纹
个人一般信息
泄露的渠道
个人信息注册时无意泄露
网上交流时被恶意爬取
保护措施 国家立法、行业自律、提升个人信息安全意识
信息社会责任
数字公民
定义 能够安全地、合法地、符合道德规范地使用数字化信息和工具的人
数字公民素养教育
尊重 数字礼仪、数字准入、数字法律
教育 数字通信、数字素养、数字商务
保护 数字权责、数字安全、数字健康
信息法规与伦理
知识产权
就信息技术领域而言，对硬件的设计制造、软件（计算机程序）的原创成果的保护
自由软件运动、开放源代码运动
自媒体时代
进行社会舆论监督、反映社情民意
网络谣言、网络诽谤
享有通信自由权、信息传播自由权、信息选择权
承担道德上的责任和义务

- 3.2 信息系统安全与防护
  - 数据加密与安全
    - 密码与秘钥
      - 目的：确保数据的保密性、完整性、不可否认性、存在性
      - 口令：登录系统需要输入的字符组合
      - 密码
        - 加密的代码
        - 密码系统
          - 明文P
          - 密码算法
            - 加密算法E
            - 解密算法D
          - 密钥
            - 加密密钥K1
            - 解密密钥K2
          - 密文C
      - 简单加密算法
        - 替代密码法：凯撒密码 'ABCD'→'CDEF'
        - 换位密码法：逆序法 'ABCD'→'DCBA'
        - 简单异或法
    - 密码体制
      - 对称（单钥）：加密密钥K1=解密密钥K2 DES算法
      - 非对称（双钥）：加密密钥K1≠解密密钥K2 RSA算法
  - 身份认证与安全
    - 身份认证
      - 用户名+口令：静态口令、动态口令（短信、口令牌）
      - 依靠生物特征识别：人脸识别、指纹识别、语音识别、虹膜认证
      - USB Key：U盾、支付盾
    - 访问控制
      - 通过用户身份及其所属策略组，限制其使用数据资源
      - 三要素：主体、客体、控制策略（授权）
  - 病毒及其防治
    - 概念：人为编制的具有破坏计算机功能或者毁坏数据，影响计算机系统的使用，并能自我复制的一组计算机指令或者程序代码
    - 特征：传染性、寄生性、隐蔽性、潜伏性、破坏性、可触发性
    - 防治
      - 防火墙、系统补丁、防病毒软件、数据备份
      - 针对手机病毒
        - 及时删除乱码信息
        - 不接受陌生请求
        - 保证下载内容的安全性
        - 不随意连接公共场合的Wi-Fi
  - 漏洞及其防护
    - 概念：一个系统存在的弱点或缺陷
    - 后门：漏洞的一种，程序编写人员为了方便进行某些调试和测试而预留的一些特权
    - 防护：防火墙、安全检测与扫描工具、使用有效的控制手段抓住入侵者、备份系统
    - 黑客
      - 一般指热衷于计算机技术或解决难题、突破限制的高手
      - 但一些人借捣乱和破坏来炫耀自己的计算机"技术"，"黑客"逐渐成为贬义词
    - 防火墙
      - 组成：服务访问规则、验证工具、包过滤、应用网关
      - 按技术分类：地址转换防火墙、数据包过滤防火墙、代理防火墙
      - 按形态分类：硬件防火墙、软件防火墙
- 4.1 ★搭建信息系统的前期准备
  - 需求分析（"做什么"）
    - 目标期待
    - 功能需求
      - 核心需求：必须要有的功能
      - 拓展需求：现有条件下可以使系统做得更令人满意的功能
      - 创新需求：不是很急迫，且一般采用比较新的技术或方法
    - 性能需求：存储容量、运行时间、传输速度、安全保密
    - 资源和环境需求
      - 硬件方面：采用什么终端设备，传感设备型号，外部设备和数据通信接口
      - 软件方面：采用什么系统软件、应用软件或用什么语言编写代码
      - 使用方面：需要使用人员的技术水平达到何种程度
    - 用户界面需求：关键是友好性 使用户能够方便、有效、愉悦地使用系统
    - 可扩展性需求：系统技术本身、业务应用
  - 可行性分析（"能不能做"）
    - 必要性
    - 可行性：技术、经济、社会意义三个方面
  - 开发模式的选择（"用什么模式做"）：C/S、B/S
  - 概要设计（"怎么做"）
    - 模块结构设计：划分系统模块、确定模块功能、决定模块间的调用关系、制定模块间的接口（数据传递）
    - 系统物理配置：硬件设备配置、应用软件的选择、通信网络的选择和设计
    - 数据库管理系统选择：数据库的性能、数据的类型、数据库管理系统平台、安全保密性能
  - 详细设计（"先干什么后干什么"）：输入设计、输出设计、人机界面设计、数据库设计、代码设计、安全设计

- 4.2 搭建信息系统
  - ★硬件搭建
    - 服务器、网络设备、传感设备和智能终端
    - 

    - 硬件搭建示意图
  - 软件开发
    - 数据管理设计
      - 数据采集　键盘、光电设备
      - 数据传输
        - 有线　双绞线、同轴电缆、光缆
        - 无线　微波、卫星、超短波和短波信道
      - 数据存储
        - 文件、数据库、云存储
        - ★数据库SQLite
          - 创建表　create table 表名(字段1 数据类型,字段2 数据类型...)
          - 插入　insert into 表名(字段名1,字段名2,...) values(值1,值2,...)
          - 删除　delete from 表名 where 字段名=某值
          - 更新　update 表名 set 字段名=新值 where字段名=某值
          - 查询　select * from 表名 where 字段名=某值
          - 基本步骤:

```
import sqlite3 #导入数据库模块
conn=sqlite3.connect("test.db") #连接到数据库文件
#如果数据库不存在，将新创建一个数据库，最后返回一个数据库对象
cur=conn.cursor() #创建游标，用于执行SQL命令
cur.execute("SELECT * FROM 表名 WHERE 字段名='xxx' ") #执行SQL命令
data=cur.fetchall() #获取查询结果的全部数据
cur.close() #关闭游标
conn.close() #关闭连接
```

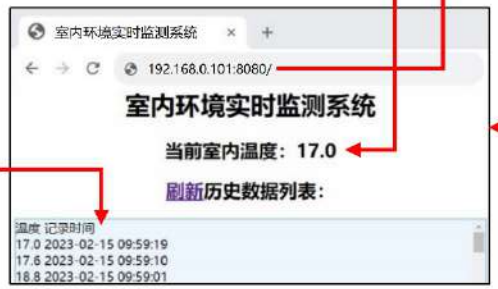
      - 数据呈现　表格、图表
    - 程序编写
      - 服务器端
        - 网站框架示意图、网站功能页面规划表　访问地址、功能说明、参数含义
        - Web服务器代码
          - sqlite3　数据库文件名xx.db、数据库路径xx/xx.db、数据表名xx
          - python占位符　print("My name is %s,I am %d years old." %(变量1,变量2))
          - JSON　一种轻量级的数据（传输）交换格式，类似于字典和列表的嵌套组合
        - Web模板文件代码
          - 模板文件名xx.html
          - Jinja2定界符
            - 表达式　{{变量或表达式}}
            - 语句　{%for i in range(10)%}
            - 注释　{#这里是注释语句#}
      - 客户端
        - B/S架构下，直接使用浏览器访问URL
        - C/S架构下，需单独编写客户端程序
      - ★智能终端
        - uart.init(...)
          - tx=pinX　将pinX设为发送端口，连接至其它设备的rx接收端口
          - rx=pinX　将pinX设为接收端口，连接至其它设备的tx发送端口
        - 物联网模块Obloq
          - 连接网络　Obloq.connectWifi(SSID,PASSWORD,timeout)
          - 设置IP地址和端口号　Obloq.httpSet(IP,PORT)
          - 发起get请求　errno,resp=Obloq.get(URL,timeout)

  - ★搭建实例
    - https://www.bilibili.com/video/BV1zg4y1p7Md
    - 附数据流向图，见下一页
  - ★故障排查　根据数据传输方向排查问题
    - 设备故障（主要）：xx设备故障
    - 连接故障（次要）：xx设备与xx设备之间的连接故障
    - 软件问题（很少）：系统缺陷，如数据库记录达到上限等
- 4.3 完善信息系统
  - ▶系统测试
    - 软件测试
      - 静态测试
      - 动态测试
      - 局限性：测试只能发现程序错误，但不能证明程序无错
      - 正确性证明　通过数学技术来确定软件是否正确，但不能发现程序中的错误和缺陷，目前还处于初级阶段
    - 硬件测试　配置检测、外观检测、硬件运行测试
    - 网络测试　配置检测、外观检测、运行测试、网络连通测试
  - 文档编写
    - 类型
      - 按阶段　系统规划阶段、系统分析阶段、系统设计阶段、系统实施阶段、系统运行和维护阶段
      - 按服务目的　用户文档、开发文档、管理文档
    - 内容　可行性研究报告、系统分析说明书、系统设计说明书、程序设计报告、系统测试报告、系统使用和维护手册、系统评价报告

信息系统中的数据流向

【Web 模板文件代码 vews.html】

```
<!doctype html>
<body><meta http-equiv="refresh" content="60"></body>
<style>
#center{ MARGIN-RIGHT: auto;MARGIN-LEFT: auto;background:#eff7ff;width:600px;
    height:500px;vertical-align:left;overflow-y:scroll; overflow-x:scroll}
</style>
<title>室内环境实时监测系统</title>
<h1 align="center">室内环境实时监测系统</h1>
<h2 align="center">当前室内温度：【{temp}】</h2>
<h2 align="center"><a href="/">刷新</a>历史数据列表：</h2>
<div id="center" style="border:2px solid #96c2f1">
  温度    记录时间<br>
  {% for i in data[::-1] %}
      {{i[2]}}    {{i[3]}} <br>
  {% endfor %}
</div>
```



【Web 服务器代码】

```
# coding= UTF-8
import json
import sqlite3
import datetime
from flask import Flask,render_template,request
DATABASE = 'data/data.db'
app = Flask(__name__)
@app.route("/")
def hello():
    db = sqlite3.connect(DATABASE)
    cur = db.cursor()
    cur.execute("SELECT * FROM sensorlog WHERE sensorid = 1")
    data = cur.fetchall()
    cur.close()
    db.close()
    temp1 = data[len(data)-1]
    temp=temp1[2]
    return render_template('vews.html', data=data,temp=temp)

#Adding data
@app.route("/input",methods=['POST','GET'])
def add_data():
    if request.method == 'POST':
        jsonval=json.loads(request.data)
        sensorid=jsonval['id']
        sensorvalue=jsonval['val']
    else:
        sensorid = int(request.args.get('id'))
        sensorvalue = float(request.args.get('val'))
    nowtime = datetime.datetime.now()
    nowtime = nowtime.strftime('%Y-%m-%d %H:%M:%S')
    db = sqlite3.connect(DATABASE)
    cur = db.cursor()
    cur.execute("INSERT INTO sensorlog(sensorid,sensorvalue,updatetime)
VALUES(%d,%f,'%s')" %(sensorid,sensorvalue,nowtime))
    db.commit()
    cur.execute("SELECT * FROM sensorlist where sensorid = %d"% sensorid)
    rv = cur.fetchall()
    cur.close()
    db.close()
    maxrv = rv[0][2]
    minrv = rv[0][3]
    if sensorvalue > maxrv or sensorvalue < minrv:
        return '1'
    else:
        return '0'

if __name__ == "__main__":
    app.run(host="0.0.0.0", port=8080, debug=True)
```

	logid	sensorid	sensorvalue	updatetime
402	437	1	18.8	2023-02-15 09:59:01
403	438	1	17.6	2023-02-15 09:59:10
404	439	1	17	2023-02-15 09:59:19

	sensorid	sensorname	maxvalue	minvalue
1	1	温度传感器	20	5
2	2	湿度传感器	80	20

【micro:bit 智能终端程序】

```
from microbit import *
import Obloq
#温度传感器（LM35）接在 pin0，蜂鸣器接在 pin8
IP="192.168.0.101"      #Web 服务器的地址
PORT="8080"             #Web 服务器的端口
SSID="jf"               #Wi-Fi 的 SSID
PASSWORD="pkq88888"     #Wi-Fi 的密码

uart.init(baudrate=115200, bits=8, parity=None, stop=1, tx=pin2, rx=pin1)
while Obloq.connectWifi(SSID,PASSWORD,10000) != True:
    display.show(".")

display.scroll(Obloq.ifconfig())
Obloq.httpSet(IP,PORT)

while True:
    temp=round((pin0.read_analog()/1024)*3000/10.24,1)
    display.scroll(temp)
    errno,resp=Obloq.get("input?id=1&val="+str(temp),20000)

    if errno == 200:
        display.show(str(resp))
        if resp == '1':
            pin8.write_digital(1)
        else:
            pin8.write_digital(0)
    else:
        display.show(str(errno))
    sleep(1000*5)
```

B①浏览器发起 HTTP 请求，并等待返回数据

B②从数据库获取数据

B③将数据填充到模板中预留的位置

B④浏览器收到服务器返回的网页文件并解释执行

A①pin0 接口从传感器读取数据

A②终端发起 HTTP 请求，并等待返回数据，该请求以 GET 方式携带了数据

A③从请求中获取数据

A④将数据存入数据库

A⑤从数据库获取数据

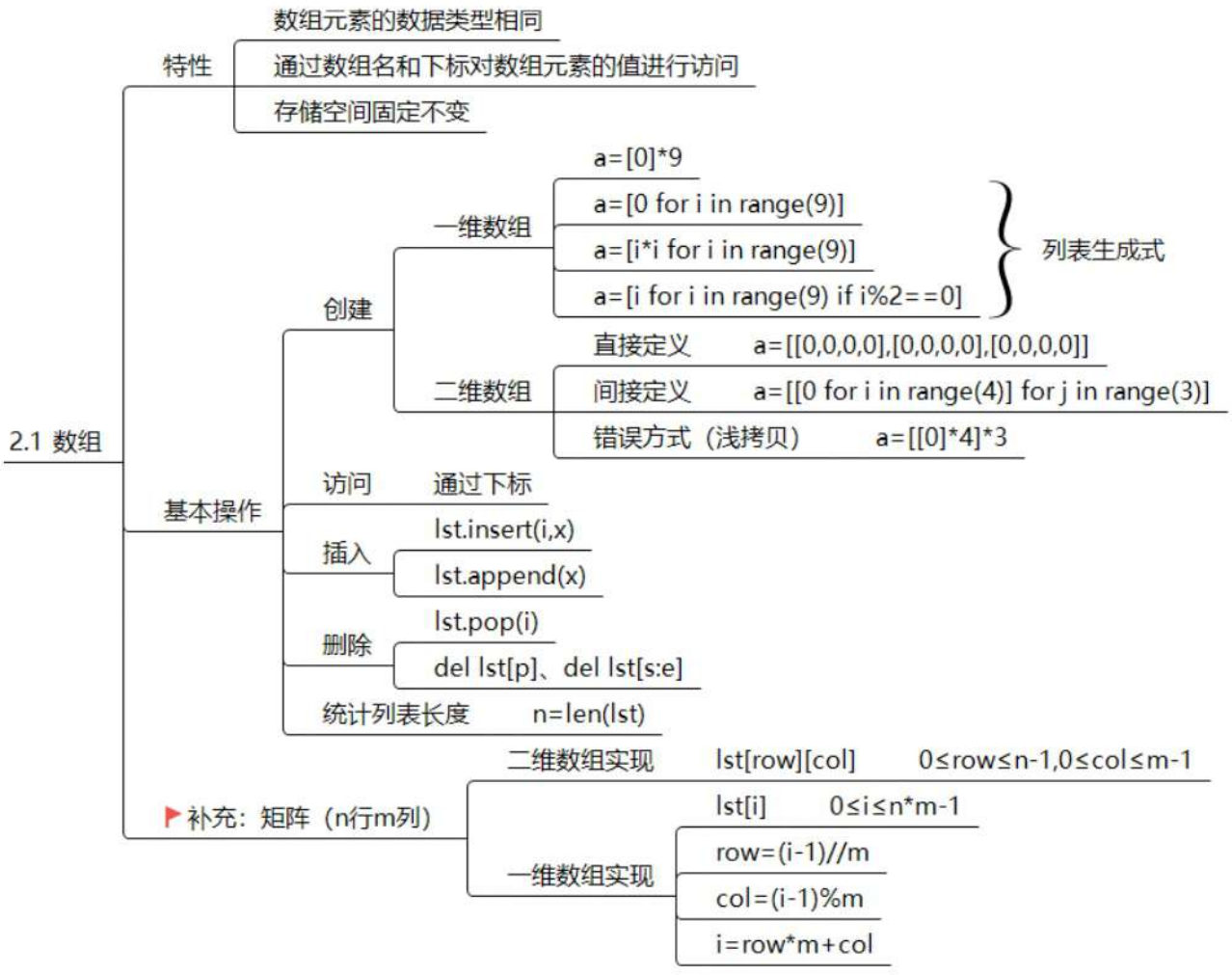
A⑥终端收到服务器返回的数据

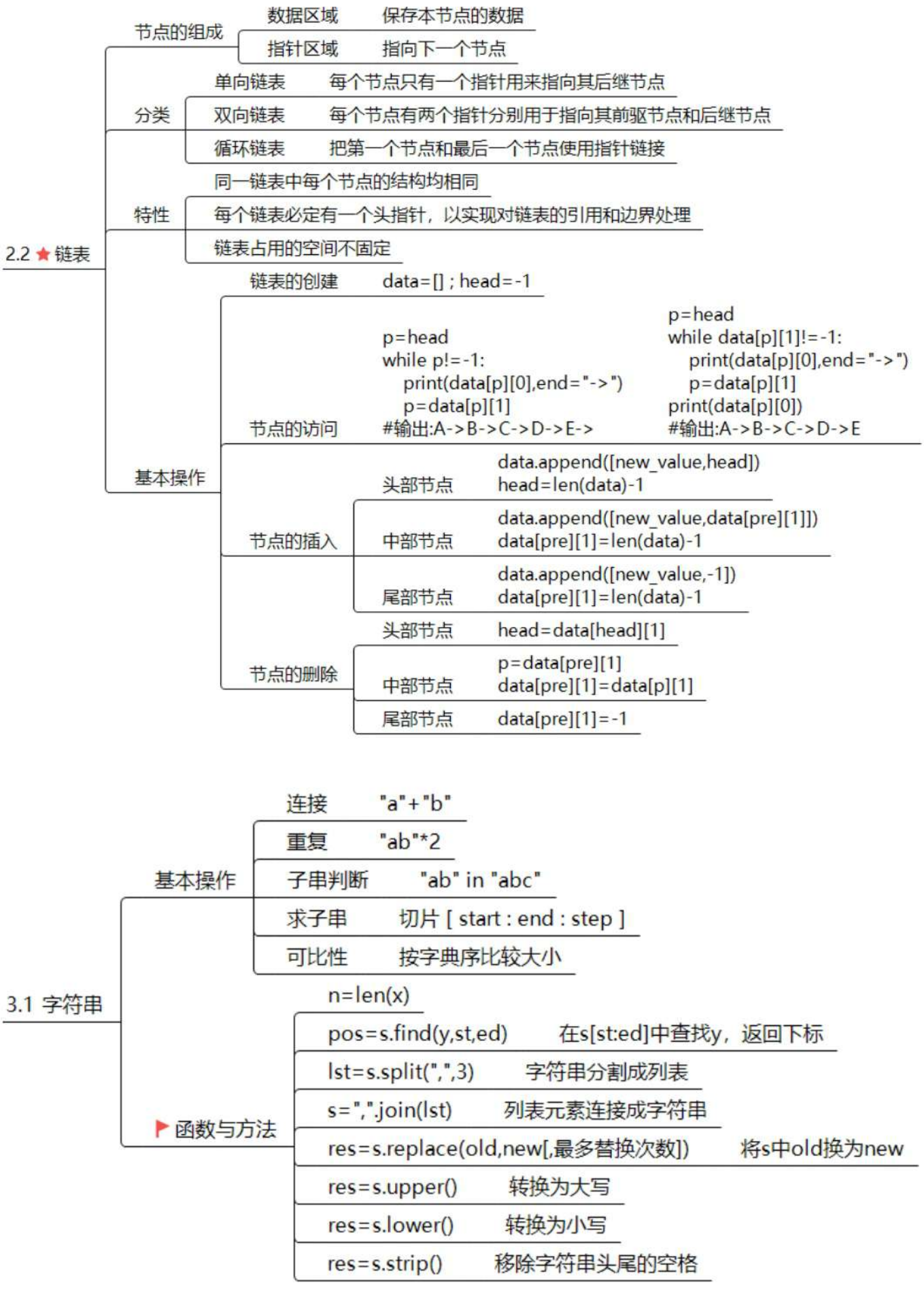
A⑦将数据发送到执行器

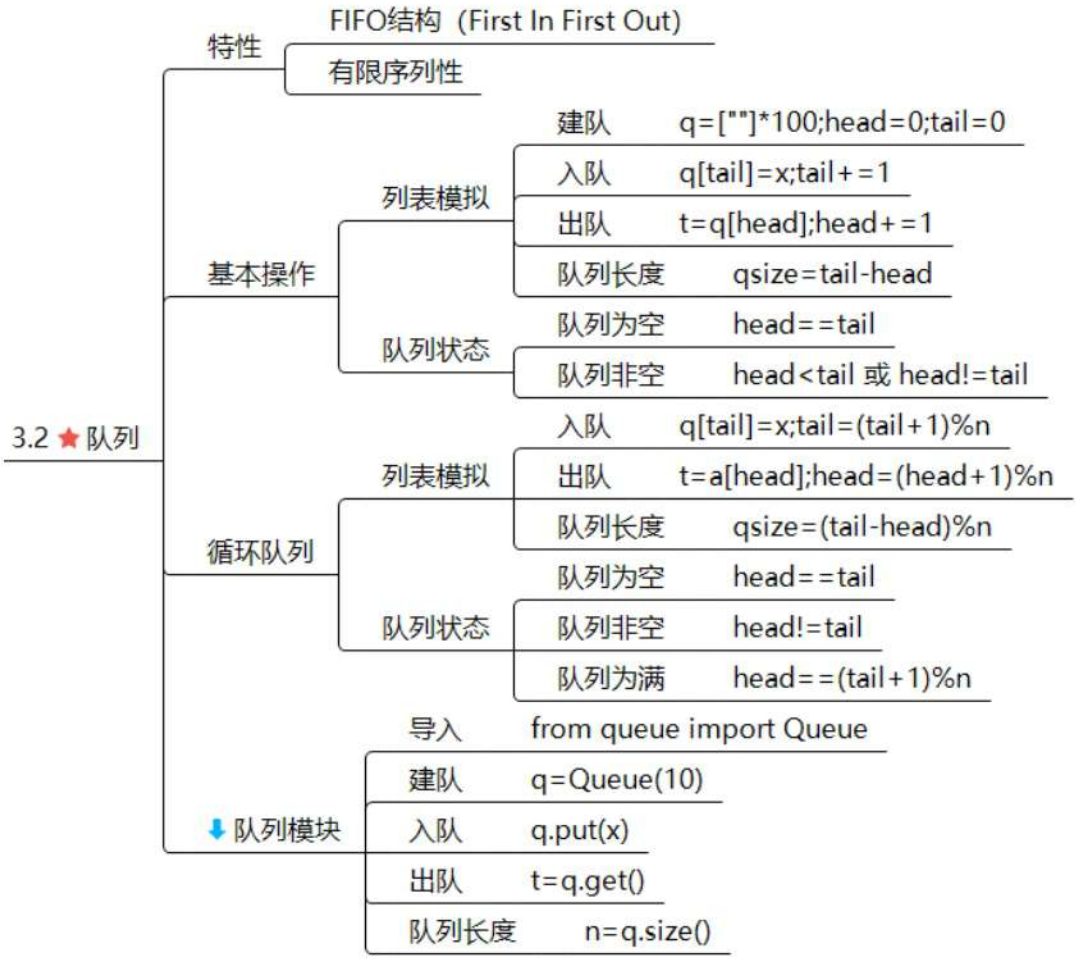
A 线路：
智能终端将传感器数据发送至服务器并存入数据库，服务器处理后返回数据给智能终端，由智能终端控制执行器。

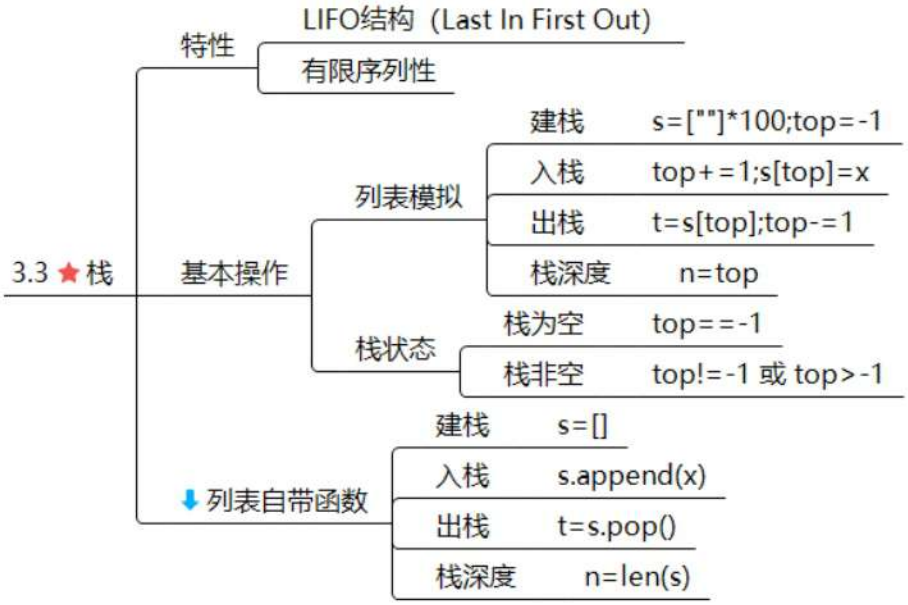
B 线路：
浏览器向服务器发起请求，服务器将数据和模板渲染后返回给浏览器，浏览器获取并显示数据。

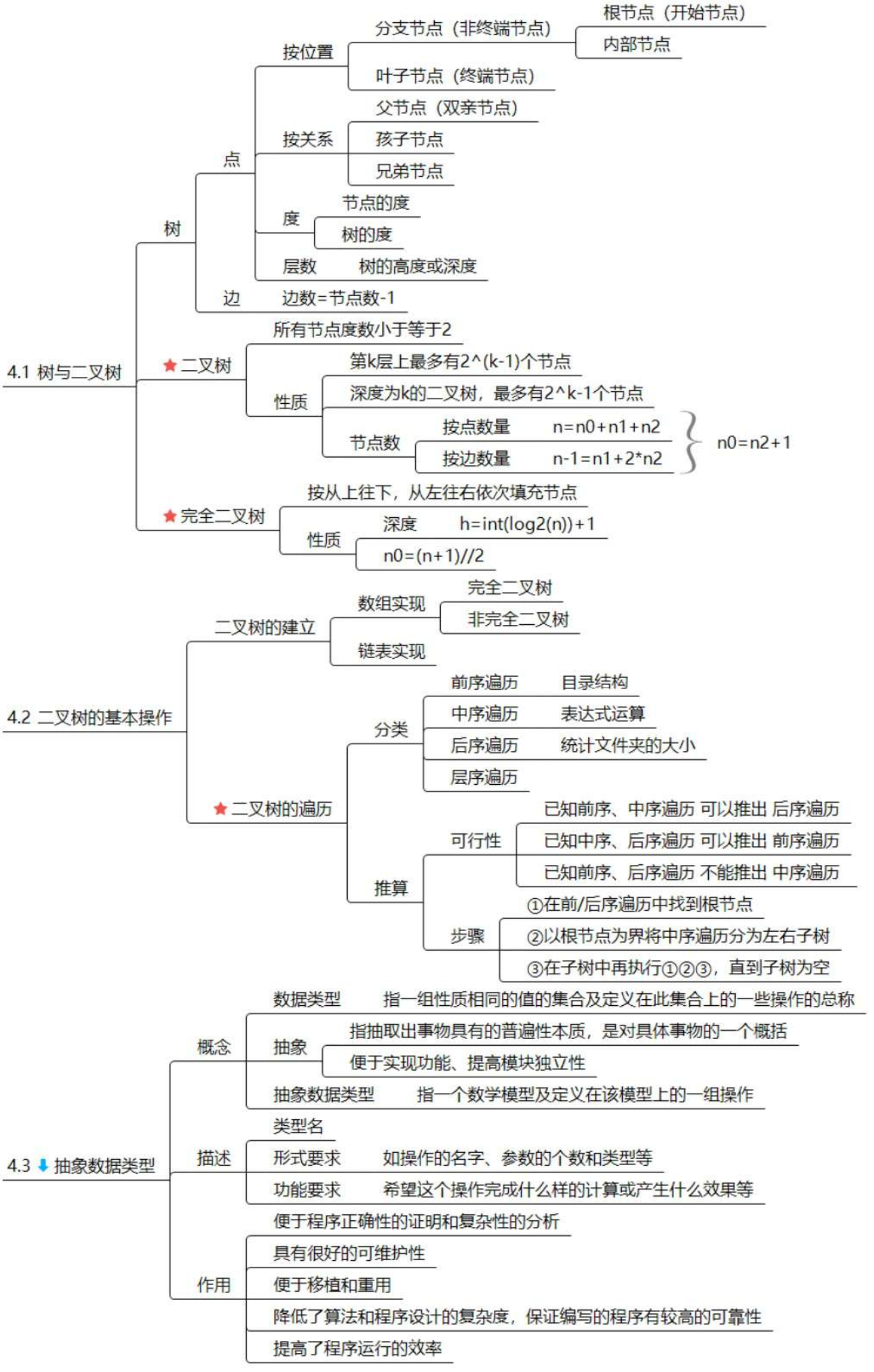
选择性必修一《数据与数据结构》

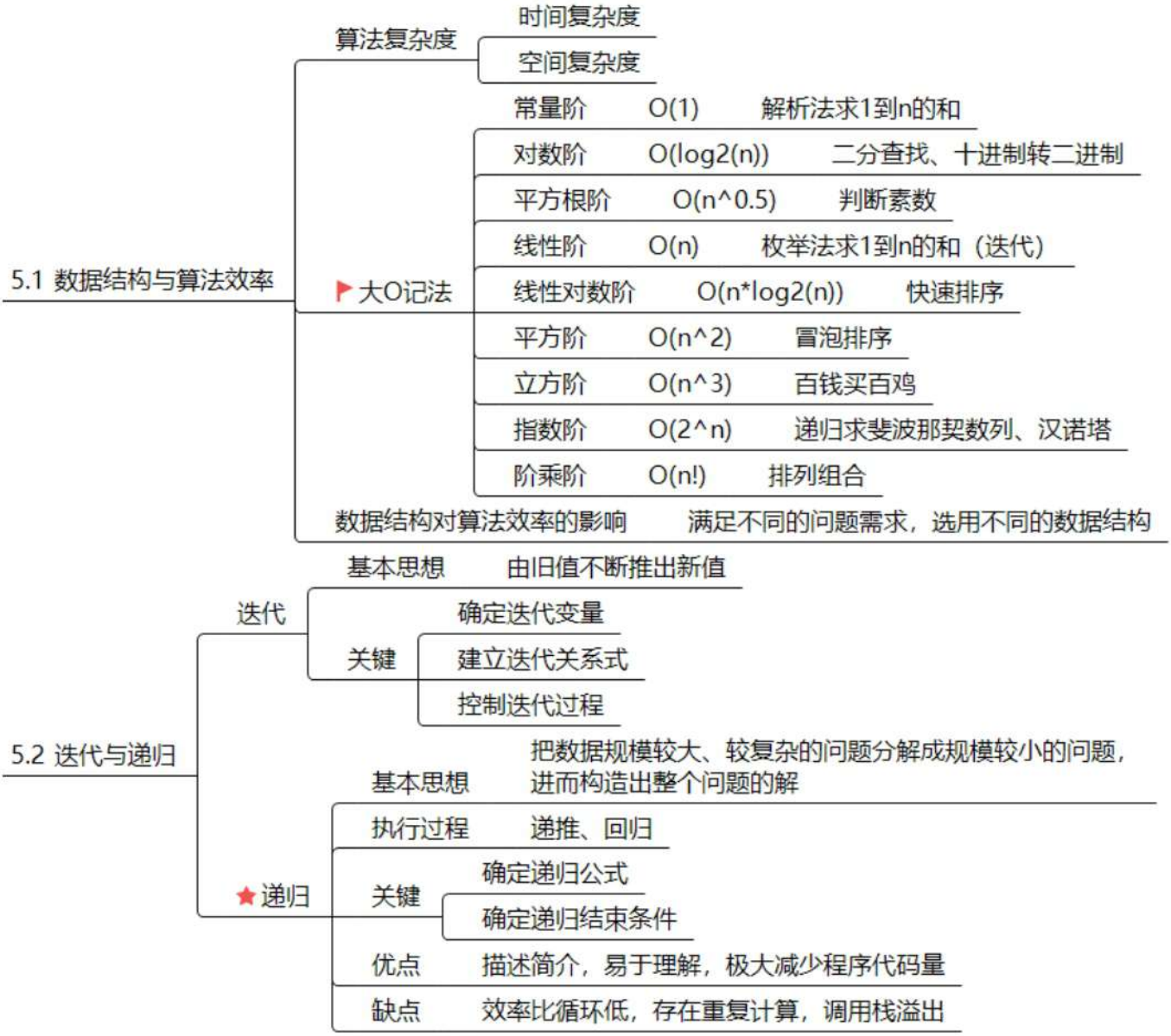


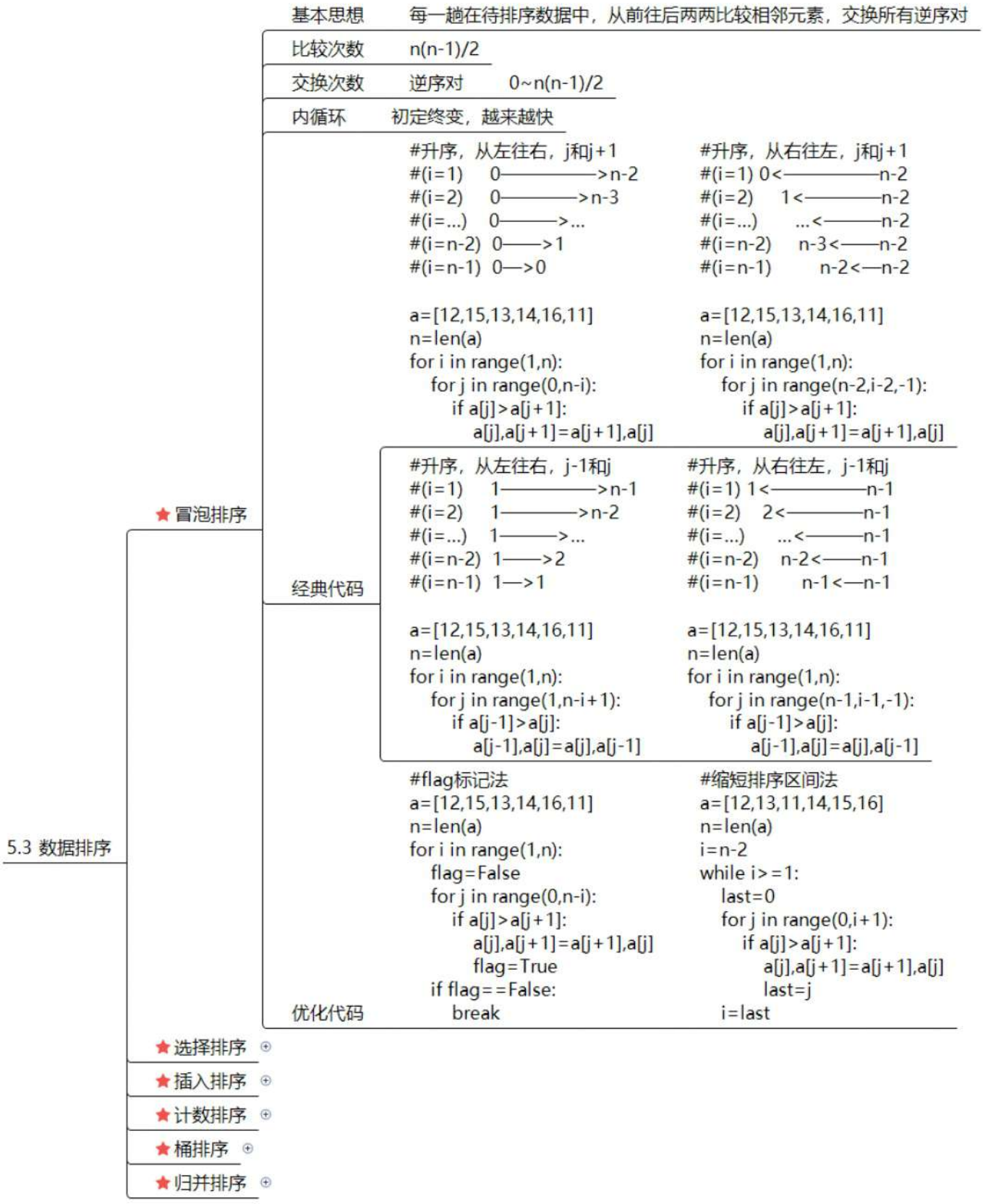

2.2 ★链表
节点的组成
数据区域 保存本节点的数据
指针区域 指向下一个节点
分类
单向链表 每个节点只有一个指针用来指向其后继节点
双向链表 每个节点有两个指针分别用于指向其前驱节点和后继节点
循环链表 把第一个节点和最后一个节点使用指针链接
特性
同一链表中每个节点的结构均相同
每个链表必定有一个头指针，以实现对链表的引用和边界处理
链表占用的空间不固定
基本操作
链表的创建 data=[] ; head=-1
节点的访问
p=head
while p!=-1:
print(data[p][0],end="->")
p=data[p][1]
#输出:A->B->C->D->E->
p=head
while data[p][1]!=-1:
print(data[p][0],end="->")
p=data[p][1]
print(data[p][0])
#输出:A->B->C->D->E
节点的插入
头部节点
data.append([new_value,head])
head=len(data)-1
中部节点
data.append([new_value,data[pre][1]])
data[pre][1]=len(data)-1
尾部节点
data.append([new_value,-1])
data[pre][1]=len(data)-1
节点的删除
头部节点 head=data[head][1]
中部节点
p=data[pre][1]
data[pre][1]=data[p][1]
尾部节点 data[pre][1]=-1
3.1 字符串
基本操作
连接 "a"+"b"
重复 "ab"*2
子串判断 "ab" in "abc"
求子串 切片 [start : end : step]
可比性 按字典序比较大小
函数与方法
n=len(x)
pos=s.find(y,st,ed) 在s[st:ed]中查找y，返回下标
lst=s.split(",",3) 字符串分割成列表
s=",".join(lst) 列表元素连接成字符串
res=s.replace(old,new[,最多替换次数]) 将s中old换为new
res=s.upper() 转换为大写
res=s.lower() 转换为小写
res=s.strip() 移除字符串头尾的空格


3.2 ★ 队列
特性
FIFO结构（First In First Out）
有限序列性
基本操作
列表模拟
建队 q=[""]*100;head=0;tail=0
入队 q[tail]=x;tail+=1
出队 t=q[head];head+=1
队列长度 qsize=tail-head
队列状态
队列为空 head==tail
队列非空 head<tail 或 head!=tail
循环队列
列表模拟
入队 q[tail]=x;tail=(tail+1)%n
出队 t=a[head];head=(head+1)%n
队列长度 qsize=(tail-head)%n
队列状态
队列为空 head==tail
队列非空 head!=tail
队列为满 head==(tail+1)%n
队列模块
导入 from queue import Queue
建队 q=Queue(10)
入队 q.put(x)
出队 t=q.get()
队列长度 n=q.size()


3.3 ★ 栈
特性
LIFO结构（Last In First Out）
有限序列性
基本操作
列表模拟
建栈 s=[""]*100;top=-1
入栈 top+=1;s[top]=x
出栈 t=s[top];top-=1
栈深度 n=top
栈状态
栈为空 top==-1
栈非空 top!=-1 或 top>-1
列表自带函数
建栈 s=[]
入栈 s.append(x)
出栈 t=s.pop()
栈深度 n=len(s)


4.1 树与二叉树
树
点
按位置
分支节点（非终端节点）
根节点（开始节点）
内部节点
叶子节点（终端节点）
按关系
父节点（双亲节点）
孩子节点
兄弟节点
度
节点的度
树的度
层数
树的高度或深度
边
边数=节点数-1
★二叉树
所有节点度数小于等于2
性质
第k层上最多有2^(k-1)个节点
深度为k的二叉树，最多有2^k-1个节点
节点数
按点数量 n=n0+n1+n2
按边数量 n-1=n1+2*n2
n0=n2+1
★完全二叉树
按从上往下，从左往右依次填充节点
性质
深度 h=int(log2(n))+1
n0=(n+1)//2
4.2 二叉树的基本操作
二叉树的建立
数组实现
完全二叉树
非完全二叉树
链表实现
★二叉树的遍历
分类
前序遍历 目录结构
中序遍历 表达式运算
后序遍历 统计文件夹的大小
层序遍历
推算
可行性
已知前序、中序遍历 可以推出 后序遍历
已知中序、后序遍历 可以推出 前序遍历
已知前序、后序遍历 不能推出 中序遍历
步骤
①在前/后序遍历中找到根节点
②以根节点为界将中序遍历分为左右子树
③在子树中再执行①②③，直到子树为空
4.3 ⬇抽象数据类型
概念
数据类型 指一组性质相同的值的集合及定义在此集合上的一些操作的总称
抽象
指抽取出事物具有的普遍性本质，是对具体事物的一个概括
便于实现功能、提高模块独立性
抽象数据类型 指一个数学模型及定义在该模型上的一组操作
描述
类型名
形式要求 如操作的名字、参数的个数和类型等
功能要求 希望这个操作完成什么样的计算或产生什么效果等
作用
便于程序正确性的证明和复杂性的分析
具有很好的可维护性
便于移植和重用
降低了算法和程序设计的复杂度，保证编写的程序有较高的可靠性
提高了程序运行的效率


5.1 数据结构与算法效率
算法复杂度
时间复杂度
空间复杂度
大O记法
常量阶 O(1) 解析法求1到n的和
对数阶 O(log2(n)) 二分查找、十进制转二进制
平方根阶 O(n^0.5) 判断素数
线性阶 O(n) 枚举法求1到n的和（迭代）
线性对数阶 O(n*log2(n)) 快速排序
平方阶 O(n^2) 冒泡排序
立方阶 O(n^3) 百钱买百鸡
指数阶 O(2^n) 递归求斐波那契数列、汉诺塔
阶乘阶 O(n!) 排列组合
数据结构对算法效率的影响 满足不同的问题需求，选用不同的数据结构
5.2 迭代与递归
迭代
基本思想 由旧值不断推出新值
关键
确定迭代变量
建立迭代关系式
控制迭代过程
递归
基本思想 把数据规模较大、较复杂的问题分解成规模较小的问题，进而构造出整个问题的解
执行过程 递推、回归
关键
确定递归公式
确定递归结束条件
优点 描述简介，易于理解，极大减少程序代码量
缺点 效率比循环低，存在重复计算，调用栈溢出


5.3 数据排序
★冒泡排序
基本思想
每一趟在待排序数据中，从前往后两两比较相邻元素，交换所有逆序对
比较次数
n(n-1)/2
交换次数
逆序对
0~n(n-1)/2
内循环
初定终变，越来越快
经典代码
#升序，从左往右，j和j+1
#(i=1) 0————>n-2
#(i=2) 0————>n-3
#(i=...) 0———>...
#(i=n-2) 0——>1
#(i=n-1) 0—>0
a=[12,15,13,14,16,11]
n=len(a)
for i in range(1,n):
for j in range(0,n-i):
if a[j]>a[j+1]:
a[j],a[j+1]=a[j+1],a[j]
#升序，从右往左，j和j+1
#(i=1) 0<————n-2
#(i=2) 1<————n-2
#(i=...) ...<————n-2
#(i=n-2) n-3<——n-2
#(i=n-1) n-2<—n-2
a=[12,15,13,14,16,11]
n=len(a)
for i in range(1,n):
for j in range(n-2,i-2,-1):
if a[j]>a[j+1]:
a[j],a[j+1]=a[j+1],a[j]
#升序，从左往右，j-1和j
#(i=1) 1————>n-1
#(i=2) 1————>n-2
#(i=...) 1———>...
#(i=n-2) 1——>2
#(i=n-1) 1—>1
a=[12,15,13,14,16,11]
n=len(a)
for i in range(1,n):
for j in range(1,n-i+1):
if a[j-1]>a[j]:
a[j-1],a[j]=a[j],a[j-1]
#升序，从右往左，j-1和j
#(i=1) 1<————n-1
#(i=2) 2<————n-1
#(i=...) ...<———n-1
#(i=n-2) n-2<——n-1
#(i=n-1) n-1<—n-1
a=[12,15,13,14,16,11]
n=len(a)
for i in range(1,n):
for j in range(n-1,i-1,-1):
if a[j-1]>a[j]:
a[j-1],a[j]=a[j],a[j-1]
优化代码
#flag标记法
a=[12,15,13,14,16,11]
n=len(a)
for i in range(1,n):
flag=False
for j in range(0,n-i):
if a[j]>a[j+1]:
a[j],a[j+1]=a[j+1],a[j]
flag=True
if flag==False:
break
#缩短排序区间法
a=[12,13,11,14,15,16]
n=len(a)
i=n-2
while i>=1:
last=0
for j in range(0,i+1):
if a[j]>a[j+1]:
a[j],a[j+1]=a[j+1],a[j]
last=j
i=last
★选择排序
★插入排序
★计数排序
★桶排序
★归并排序

5.3 数据排序

★冒泡排序

★选择排序

基本思想：每一趟从待排序数据中选出最小的元素，放到已排序序列的末尾

比较次数：n(n-1)/2

交换次数：0~n-1

内循环：将未排序数据全部遍历覆盖到

#升序，从左往右

```
#(i=0)   1————————>n-1
#(i=1)     2————————>n-1
#(i=...)     ...————————>n-1
#(i=n-3)     n-2——>n-1
#(i=n-2)       n-1—>n-1
```

#升序，从右往左

```
#(i=1)    0<————————n-2
#(i=2)    0<————————n-3
#(i=...)  0<————————...
#(i=n-2)  0<——1
#(i=n-1)  0<—0
```

经典代码

```
a=[12,15,13,14,16,11]
n=len(a)
for i in range(0,n-1):
    k=i
    for j in range(i+1,n):
        if a[j]<a[k]:
            k=j
    if k!=i:
        a[i],a[k]=a[k],a[i]
```

```
a=[12,15,13,14,16,11]
n=len(a)
for i in range(1,n):
    k=n-i
    for j in range(n-i-1,-1,-1):
        if a[j]>a[k]:
            k=j
    if k!=n-i:
        a[n-i],a[k]=a[k],a[n-i]
```

★插入排序

基本思想：每一趟从待排序数据中选出第一个元素，插入到已排序序列中

比较次数：n-1~n(n-1)/2

交换(移动)次数：0~n(n-1)/2

内循环：初变终定，越来越慢

#升序，从右往左

```
#(i=1)    0<—0
#(i=2)    0<——1
#(i=...)  0<————...
#(i=n-2)  0<————————n-3
#(i=n-1)  0<————————n-2
```

#升序，从左往右

```
#(i=1)          n-1—>n-1
#(i=2)        n-2——>n-1
#(i=...)      ...————>n-1
#(i=n-2)    2————————>n-1
#(i=n-1)  1————————>n-1
```

经典代码

```
a=[12,15,13,14,16,11]
n=len(a)
for i in range(1,n):
    k=a[i]
    j=i-1
    while j>=0 and a[j]>k:
        a[j+1]=a[j]
        j=j-1
    a[j+1]=k
```

```
a=[12,15,13,14,16,11]
n=len(a)
for i in range(1,n):
    k=a[n-i-1]
    j=n-i
    while j<=n-1 and a[j]<k:
        a[j-1]=a[j]
        j=j+1
    a[j-1]=k
```

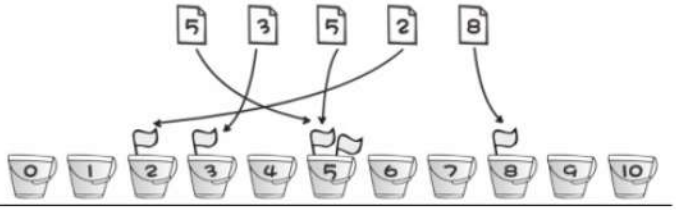
★计数排序

★桶排序

★归并排序

5.3 数据排序

- ★冒泡排序
- ★选择排序
- ★插入排序
- ★计数排序
  - 基本思想：对每个数据计数，并将数据的计数信息用于排序



  - 经典代码

```
import random
m,n=10,5
a=[random.randint(0,m) for i in range(n)]
cnt=[0]*(m+1)
for i in range(n):
    cnt[a[i]]+=1
for i in range(0,m+1):
    for j in range(cnt[i]):
        print(i,end=" ")
```

- ★桶排序
  - 基本思想：将数据划分到各个桶中，然后对每个桶进行排序，最后把排序后的桶合并在一起
- ★归并排序
  - 基本思想：分治思想：将序列中相邻两个数字进行归并操作，将归并后的相邻序列再次进行归并操作，一直重复，直到归并成只剩一个序列
  - 经典代码

```
import random
a=[random.randint(1,99) for i in range(10)]
def merge(a,L,M,R):    #将a[L:M+1]与a[M+1:R+1]合并
    temp=[]
    i=L;j=M+1
    while i<=M and j<=R:
        if a[i]<a[j]:
            temp.append(a[i]);i+=1
        else:
            temp.append(a[j]);j+=1
    while i<=M:
        temp.append(a[i]);i+=1
    while j<=R:
        temp.append(a[j]);j+=1
    for i in range(len(temp)):
        a[L+i]=temp[i]

def sort(a,lft,rgt):    #将大问题分解为两个更小的问题
    if lft<rgt:
        mid=(lft+rgt)//2
        sort(a,lft,mid)
        sort(a,mid+1,rgt)
        merge(a,lft,mid,rgt)

sort(a,0,len(a)-1)
print(a)
```

5.4 数据查找

顺序查找

- 基本思想：又称线性查找；从顺序表的一端开始，依次将每个元素的关键字和给定值进行比较
- 查找次数：1~n

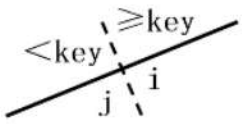
★二分查找

- 基本思想：又称折半查找、对分查找；将查找键与有序数组内处于中间位置的元素进行比较，若不同，则再确定在数组的前半部分还是后半部分继续查找，直到获得最终结果
- 查找次数：1~int(log2(n))+1
- 经典代码：

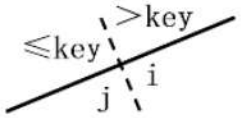
```
a=[1,4,7,10,15,20,24,30,36,99]
key=int(input())
i,j=0,len(a)-1
pos=-1
while i<=j:
    m=(i+j)//2
    if a[m]==key:
        pos=m
        break
    elif a[m]>key:
        j=m-1
    else:
        i=m+1
```

- 查找边界：在有序数组中，查找符合条件的数据第一次或最后一次出现的位置

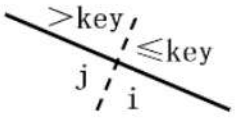
```
if a[m]<key:
    i=m+1
else:
    j=m-1
```



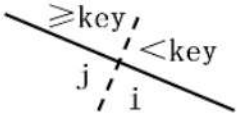

```
if a[m]<=key:
    i=m+1
else:
    j=m-1
```

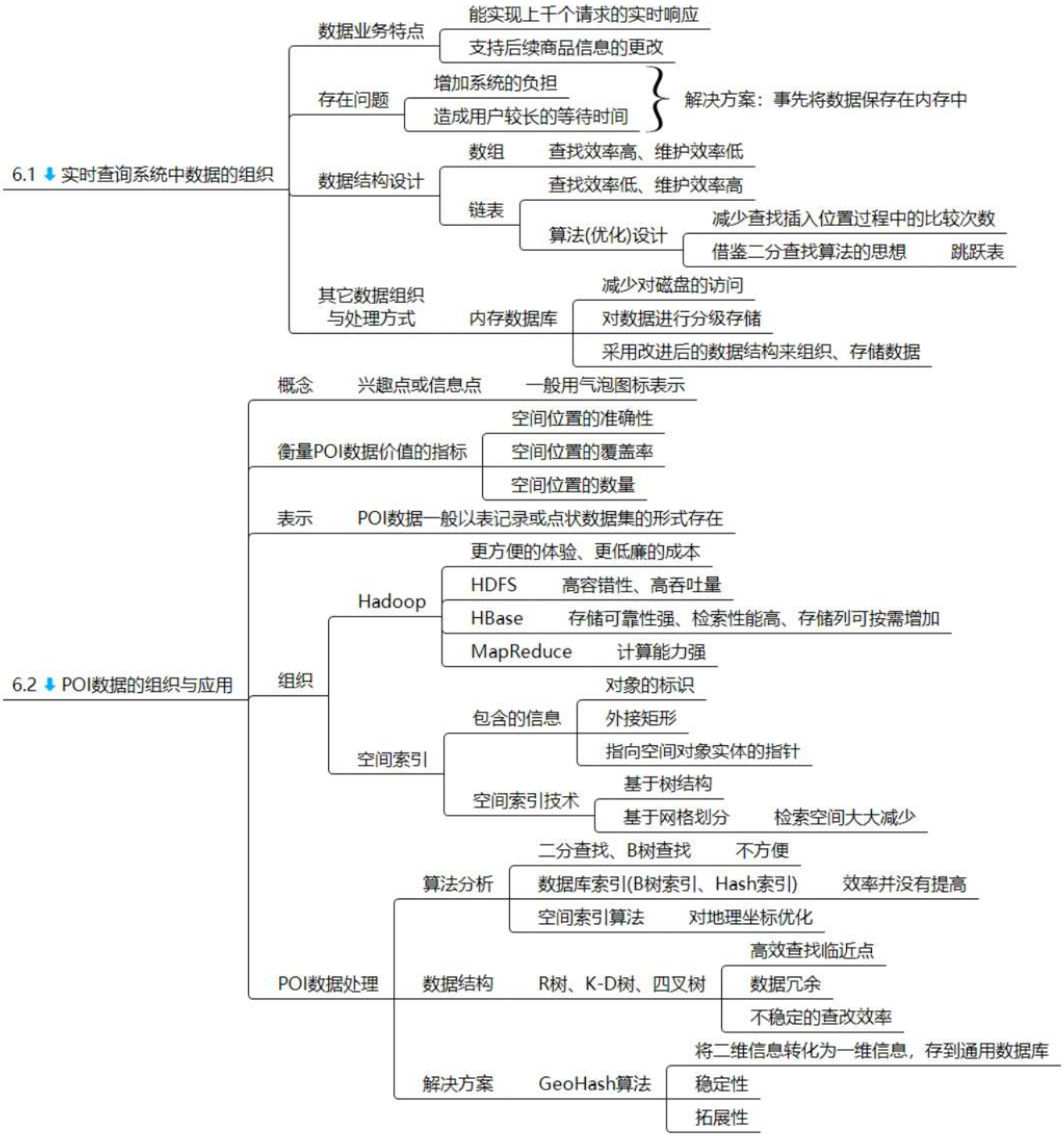



```
if a[m]>key:
    i=m+1
else:
    j=m-1
```




```
if a[m]>=key:
    i=m+1
else:
    j=m-1
```




6.1 实时查询系统中数据的组织
数据业务特点
能实现上千个请求的实时响应
支持后续商品信息的更改
存在问题
增加系统的负担
造成用户较长的等待时间
解决方案：事先将数据保存在内存中
数据结构设计
数组
查找效率高、维护效率低
链表
查找效率低、维护效率高
算法(优化)设计
减少查找插入位置过程中的比较次数
借鉴二分查找算法的思想
跳跃表
其它数据组织与处理方式
内存数据库
减少对磁盘的访问
对数据进行分级存储
采用改进后的数据结构来组织、存储数据
6.2 POI数据的组织与应用
概念
兴趣点或信息点
一般用气泡图标表示
衡量POI数据价值的指标
空间位置的准确性
空间位置的覆盖率
空间位置的数量
表示
POI数据一般以表记录或点状数据集的形式存在
组织
Hadoop
更方便的体验、更低廉的成本
HDFS
高容错性、高吞吐量
HBase
存储可靠性强、检索性能高、存储列可按需增加
MapReduce
计算能力强
空间索引
包含的信息
对象的标识
外接矩形
指向空间对象实体的指针
空间索引技术
基于树结构
基于网格划分
检索空间大大减少
POI数据处理
算法分析
二分查找、B树查找
不方便
数据库索引(B树索引、Hash索引)
效率并没有提高
空间索引算法
对地理坐标优化
数据结构
R树、K-D树、四叉树
高效查找临近点
数据冗余
不稳定的查改效率
解决方案
GeoHash算法
将二维信息转化为一维信息，存到通用数据库
稳定性
拓展性

附：信息技术考前抢分秘籍（★重难点、▲易错点）

——滚烫的青春，我没有天分，还想倔强，不留遗憾一分

一、数据与信息

1. ▲信息不会产生损耗，也**不能脱离载体**，但**可以脱离它所反映的事物被存储、保存和传播**
2. ▲知识更接近“应用与行动”，智慧更接近“认知与看法”
3. ▲位/比特/bit 和 字节/Byte 的换算（**1Byte = 8bit**）
4. ▲WAV 容量 = **采样频率 * 量化位数 * 声道数 * 时间 / 8**
5. ★颜色模式和位深度（x 位色=2^x色）：RGB 24b、256 级灰度 8b、256 色 8b、黑白图 1b
6. 视频制式：PAL 制式（25fps）、NTSC（30fps），计算视频容量时，一般忽略声音的容量

二、算法的程序实现（Python）

1. ▲表达式运算优先级： ** 优先于 * / // % 优先于 + -
2. ▲判断是否相等时，是双等于号 ==
3. ★切片用冒号，如 s[1:9:2]，s[1:9]，s[1::2]，s[:9:2]，s[::2]，s[1::]，s[:9:]，s[::]
4. ▲range 用逗号，如 range(1,9,2)，range(1,9)，range(9)；错误：range(1,,2)
5. ▲**遍历列表时，不要漏写 range**，如 for i in range(len(a))
6. ▲int 函数是去掉小数，保留整数；//是将结果向下取整
7. 自定义函数可以返回多个值，调用时用多个变量接收，如 a,b=myzdy(x)
8. ▲计算数量、位置时要考虑**错一问题**

三、数据处理与应用（大数据、pandas、matplotlib）

1. ▲异常数据：发生概率较小，可能是要去除的不重要的数据，也可能是含有重要信息的数据，如技术考了 99 分（平常考 80 左右）
   ▲逻辑错误：违背业务规则和逻辑，如技术考了 101 分
2. ★大数据特征（4V）、大数据思维（全体数据、混杂性、相关性）、**大数据处理（分治思想）**
3. ▲图数据，指的是拓扑图（Graph），是一种以点和边来表示实体和关系的数据结构，而不是图像数据（Image/Picture）
4. ★DataFrame 中取单个值的 3 种方法：
   a） print(df.at[1,"分数"]) #**行列法**：先 index 再 column
   b） print(df["分数"][1])或 print(df.分数[1]) #**列行法**：先 column 再 index
   c） print(df.values[4][2]) #**二维数组法**：先行再列
   d） 循环是 for i in df.index 时，优先用“行列法”、“列行法”取值
   e） 循环是 for i in range(len(df))时，优先用“二维数组法”取值
5. ▲DataFrame 筛选时，不能对列进行额外的操作，错例：df=df[df.date[5:7]=="11"]
6. ▲DataFrame 中修改单个值，只能用 df.at[行索引,"列标题"]=xxx
7. ★考场上，要时刻呈现 df 对象目前长什么样？

如执行 df=df.groupby("xxx",as_index=True).count()之后，df 中的数据及组织形式

8. ▲求 df 长度时，不能用 n=df.count()，要用 n=len(df)
9. ▲使用 groupby 时一定要留意 as_index 的值
10. ▲使用 sort_values 时一定要留意 ascending 的值
11. ▲DataFrame 中除了 insert 以外，其它**所有函数都不会更新原数据对象，需要重新赋值**
12. ▲matplotlib 中，显示绘制的图例，需要执行 plt.legend()
13. ▲绘制水平柱形图（barh）时，纵轴（x 轴）上的数据，要从原点朝远离原点的方向看

四、人工智能

1. ▲图灵机与图灵测试的区别：
   a) 图灵机：一种计算机制，与“原始递归函数”、“lambda 演算”等效
   b) 图灵测试：测试机器是否具有智能的一种方法
2. ★三大主义
   a) “**符号主义**”关键词：逻辑学派，心理学派，计算机学派，基于规则，符号的推理和运算，手工构造知识库+推理引擎，知识工程，专家系统，深蓝（象棋）、沃森（问答）
   b) “**联结主义**”关键词：仿生学派，生理学派，数据驱动，神经元，深度学习，图像分类，xx 识别（语音、文字、图像、人脸、指纹、虹膜等），AlphaGo（围棋），机器翻译、无人驾驶
   c) “**行为主义**”关键词：进化主义，控制论学派，问题引导，交互-反馈，试错学习，强化学习，AlphaGo Zero（围棋）
3. ▲三大应用：领域人工智能、跨领域人工智能、混合增强智能的区别
4. ▲注意：扫描识别条形码和二维码，不属于人工智能

五、信息系统（概述、支撑技术、安全、搭建）

1. ▲信息技术发展形态：以**计算机**为核心→以**互联网**为核心→以**数据**为核心
2. ★信息系统的局限性：环境依赖性（最大）、本身安全隐患、加剧数字鸿沟
3. ▲服务器属于硬件，是计算机的一种，性能比普通计算机更好
4. ▲数据库，既不是硬件也不是软件。数据库是数据的集合，是软件的一部分，通常以文件形式呈现。（课本 P40 软件的定义：在计算机上运行的程序及其数据和文档的总和）
5. ▲控制器和执行器（输出设备）的区别，不要搞混
6. ★HTML（超文本标记语言，用于编写网页文件）和 HTTP（超文本传输协议，用于传输网页文件）
7. ▲网络三大功能：数据通信（最基本）、资源共享、分布处理
8. ▲网络三大分类：计算机网络（LAN、MAN、WAN）、移动通信网络、广播电视网络
9. ▲网络三大组成：计算机系统、数据通信系统、网络软件和网络协议
10. ▲网关，是一个虚拟的概念，一般在路由器中
11. ▲仅凭“客户端”3 字不能判定架构类型，B/S：通用客户端（浏览器）；C/S：专用客户端
12. ▲保护存储介质的安全（磁盘阵列、数据备份、异地容灾）
    ▲保护数据本身的安全（**加密-保密性；数据校验-完整性；数字签名-不可否认性**）

13. ▲身份认证（用户访问权限）≠访问控制（用户操作权限）
14. ▲端口的 2 种默认：
    a） Flask 框架在启动 Web 应用服务端程序时，若不设定端口值，则默认为 5000
    b） 书写 URL 时，端口若为 80，URL 中可以省略不写
15. ★▲信息系统中各端的通信数据匹配
    a） 路由：**开头要加/，后面绝对不能加参数**，如@app.route("/input")
    b） 视图函数：当收到的 http 请求和路由匹配上时，和该路由绑定的视图函数会被调用
    c） 浏览器发起的 http 请求：是完整的 URL，如 http://192.168.1.10:8080/show?id=1
    d） 智能终端发起的 http 请求：请求中的**协议、IP、端口已单独设置，此处省略不写**，**开头不加/**，可以带参数，如 0bloq.get("input?id=1&val="+str(tmp),20000)
    e） **从请求中获取的参数都是字符串型的，必要时需进行类型转换**
16. ★▲信息系统中的数据流向，搞清楚“数据是哪个，数据从哪来，数据到哪去”
17. ★▲前期准备中，注意区分某些相似步骤

a） 采用何种型号硬件设备	∈需求分析	|	硬件设备配置	∈概要设计
b） 用户界面要友好	∈需求分析	|	人机界面设计	∈详细设计
c） 选择数据库管理系统	∈概要设计	|	数据库设计	∈详细设计

18. ★开放性问题答题思路：
    a） 故障排查类：①硬件（设备） ②网络（线路） ③软件（阈值、程序逻辑、数据库）
    b） 性能提升类：按计算机五大部件组成分析（冯·诺依曼结构）
    c） 数据异常类：按数据流向分析（传感器→智能终端↔服务器↔数据库↔服务器↔浏览器）
    执行器↙

六、数据结构

1. ▲条件判断语句中，在对列表元素、链表节点引用时，一定要**考虑下标是否会越界**
2. ★队列状态判断
    a） 普通队列：长度 tail-head、队空 head==tail、非空 head<tail
    b） 循环队列：长度(tail-head)%len(que)、队空 head==tail、非空 head!=tail、队满 (tail+1)%len(que)==head
3. ▲二叉树中，n=n0+n1+n2，n-1=n1+2*n2 → **n0=n2+1**
4. ★二叉树的**第 k 层最多有 2^(k-1)个节点**；**深度为 k 的二叉树最多有 2^k-1 个节点**
5. ★会用大 0 记法进行算法的复杂度分析，熟悉经典算法的复杂度
6. ★迭代算法与递归算法的比较，如斐波那契数列的两种实现方式的时间复杂度比较
7. ★掌握基础排序算法（冒泡、选择）的多种形态（正向/逆向、升序/降序、比较的元素）
8. ★冒泡排序中：比较次数 n*(n-1)/2≥交换次数=**逆序数**
9. ★对 N 个数进行对分查找，最多查找次数=最深层数=**int($\log_2 N$)+1**
10. ★掌握**基础对分查找的边界问题**

七、答题规范与细节

1. 用笔读题，圈划要点。1~6 题题干基本不用看，直接看选项
2. 选择题（1）否定词、极端字词、排除法（2）选项看全（3）答案分布较均匀

3. 用字典记法取值时，列名要完整，不能漏写单位，且一定要加双引号
4. 不要漏写下引号、右中括号、右小括号
5. 要写的是数据库名（文件名）还是数据表名
6. 注重审题：单选 or 多选？填文字 or 字母？程序是正向加密 or 逆向解密？
7. 切忌出现错别字、拼写错误（True/False）、语法错误，书写端正规范
8. 每个 if 开头的代码块都是独立的，都要执行
9. 切片的分隔符是冒号，range 的分隔符是逗号
10. 输入输出要记得考虑是否加 int 和 str，ASCII 字符转换用 ord 和 chr
11. =还是==还是!=，这些特殊的符号要重视，一旦填错就是全错
12. 特殊值检验，极限（边界）检验
13. 答题纸上只填写划线部分：room=**room+1**、**room=room+1**
14. 答题区域：程序的填空和改错的答题顺序千万不能错

八、程序大题常见挖空位置以及填空逻辑

(1)通过观察变量的命名、题干、注释，猜测和理解的变量的含义

t tmp temp ans res / f flag c cnt count n num s sum / l lft left r rgt right / s st start b bg begin e ed end / q que queue s st stack lst list p pos p pre

(2)变量、数组的初始化（flag=True、total=0）

(3)循环开始前，看循环变量或循环条件是否需要初始化赋值，通常可根据(4)(5)推理

while (4)边界检测（是否超出实际范围 i<=n）、(5)是否满足题设条件（not flag）：

(6)每趟循环开始时，考虑数组下标与循环变量的对应关系，可能需要处理和变换

(7)若有自定义函数，考虑是否需要调用自定义函数（ans=getPos(i)）

if (8)满足题设条件（某一种情况）：

(7)按需调用自定义函数

(9)看循环变量、关键数组、关键变量是否需要更新（累加、计数、约瑟夫环...）

(10)若代码对称，可以模仿填写，但一般会有小变化，往往在题目中有说明和体现

elif (8)满足题设条件（另一种情况）：

同(9)(10)

else （剩下的情况）：

同(9)(10)

每趟循环结束前，

(11)看是否需要用到前面 if 结构中处理的结果（更新极值、累加器等）

(12)看是否需要输出变量（可以联系图中的输出结果）

(13)循环变量的改变（更新、重置、复位、累加、计数、约瑟夫环、字符串连接）

(14)循环结束后，是否需要对最后一次循环作特殊补充处理（如最后一组数据尚未处理）

(15)程序结束前，注意结合题目要求，完成对结果的处理，包括结果的运算和类型的转换

```
def getPos(p):
    (16)可以有返回值，也可以不返回，根据实际作用决定
    return x
```